# ornl

AD-A272 125

# OAK RIDGE
# NATIONAL
# LABORATORY

**MARTIN MARIETTA**

## Proceedings of the Seventh International Symposium on Methodologies for Intelligent Systems (Poster Session)

June 15–18, 1993
Trondheim, Norway

DTIC
ELECTE
NOV 0 1 1993
S A D

Karen S. Harber, Editor

93-26255

93 10 034

Engineering Physics and Mathematics Division

# PROCEEDINGS OF THE SEVENTH INTERNATIONAL SYMPOSIUM ON METHODOLOGIES FOR INTELLIGENT SYSTEMS (POSTER SESSION)

June 15–18, 1993
Trondheim, Norway

Karen S. Harber, Editor
Center for Engineering Systems Advanced Research
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, TN 37831-6364

DATE PUBLISHED — May 1993

# Preface

This volume contains papers which were selected for presentation at the Poster Session of the Seventh International Symposium on Methodologies for Intelligent Systems - ISMIS'93, held in Trondheim, Norway, June 15-18, 1993. The symposium was hosted by the Norwegian Institute of Technology and sponsored by The University of Trondheim, NFR/NTNF – The Norwegian Research Council, UNC-Charlotte, Office of Naval Research, Oak Ridge National Laboratory and ESPRIT BRA Compulog Network of Excellence.

ISMIS is a conference series that was started in 1986 in Knoxville, Tennessee. It has since then been held in Charlotte, North Carolina, once in Knoxville, and once in Torino, Italy.

The Organizing Committee has decided to select the following major areas for ISMIS'93:

1. Approximate Reasoning
2. Constraint Programming
3. Expert Systems
4. Intelligent Databases
5. Knowledge Representation
6. Learning and Adaptive Systems
7. Manufacturing
8. Methodologies

The contributed papers were selected from more than 120 full draft papers by the following Program Committee: Jens Balchen (NTH, Norway), Alan W. Biermann (Duke, USA), Alan Bundy (Edinburgh, Scotland), Jacques Calmet (Karlsruhe, Germany), Jaime Carbonell (Carnegie-Mellon, USA), David Hislop (US Army Research Office), Eero Hyvonen (VTT, Finland), Marek Karpinski (Bonn, Germany), Yves Kodratoff (Paris VI, France), Jan Komorowski (NTH, Norway), Kurt Konolige (SRI International, USA), Catherine Lassez (Yorktown Heights, USA), Lennart Ljung (Linköping, Sweden), Ramon Lopez de Mantaras (CSIC, Spain), Alberto Martelli (Torino, Italy), Ryszard Michalski (George Mason, USA), Jack Minker, (Maryland, USA), Rohit Parikh (CUNY, USA), Judea Pearl (UCLA, USA), Don Perlis (Maryland, USA), Francois G. Pin (ORNL, USA), Henri Prade (Toulouse, France), Zbigniew W. Raś (UNC, USA), Barry Richards (Imperial College, UK), Colette Rolland (Paris I, France), Lorenza Saitta (Trento, Italy), Erik Sandewall (Linköping, Sweden), Richmond Thomason (Pittsburgh, USA), Enn Tyugu (KTH, Sweden), Ralph Wachter (ONR, USA), S. K. Michael Wong (Regina, Canada), Erling Woods (SINTEF, Norway), Maria Zemankova (NSF, USA) and Jan Zytkow (Wichita State, USA). Additionally, we acknowledge the help in reviewing the papers from: M. Beckerman, Sanjiv Bhatia, Jianhua Chen, Stephen Chenoweth, Bill Chu, Bipin Desai, Keith Downing, Doug Fisher, Melvin Fitting, Theresa Gaasterland, Atillio Giordana, Charles Glover, Diana Gordon, Jerzy Grzymala-Busse, Cezary Janikow, Kien-Chung Kuo, Rei-Chi Lee, Charles Ling, Anthony Maida, Stan Matwin, Neil Murray, David Mutchler, Jan Plaza, Helena Rasiowa, Steven Salzberg, P. F. Spelt, David Reed, Michael Sobolewski, Stan Szpakowicz, Zbigniew Stachniak, K. Thirunarayan, Marianne Winslett, Agata Wrzos-Kamińska, Jacek Wrzos-Kamiński, Jing Xiao, Wlodek Zadrozny and Wojtek Ziarko.

# TABLE OF CONTENTS

v

# POSTER SESSIONS

## APPROXIMATE REASONING

"An Application of Rough Sets in Knowledge Synthesis," S. K. M. Wong (Univ. of Regina, Canada), Y. Y. Yao (Lakehead Univ., Canada), and L. S. Wang (Univ. of Regina, Canada)

## EXPERT SYSTEMS

"Formalization of an Ontology of Ceramic Science in CLASSIC," Piet-Hein Speel, Paul E. van der Vet, Wilco ter Stal, and Nicolaas J. I. Mars (Univ. Twente, The Netherlands)

## INTELLIGENT DATABASES

"A Natural Language Generation System for a Heterogeneous Distributed Database System," Bipin C. Desai and Georgios Ioanni Kouklakis (Concordia Univ., Canada)

"A Relational Model for Imprecise Queries," Weining Zhang (Univ. of Lethbridge, Canada), Clement Yu (Univ. of Illinois-Chicago, USA), Gaoming Wang, Tracy Pham (Omron, Japan), and Hiroshi Nakajima (Alberta, Canada)

## LEARNING AND ADAPTIVE SYSTEMS

"A Self-Learning Bayesian Expert System (B.E.St.)," Farrokh Alemi, Pallav Bhatt, Eric Eisenstein, Adam Fadlalla, Richard Stephens, and John Butts (Cleveland State Univ., USA)

"Strategy Acquisition by an Artificial Neural Network: Experiments in Learning to Play a Stochastic Game," Neal M. Mazur (Union College, USA)

"Multivariate Discretization of Continuous Attributes for Machine Learning," Thomas W. Rauber (Univ. of Nova de Lisboa, Portugal), Dinu Coltuc (ICPE, Romania), and Adolfo S. Steiger-Garção (Univ. of Nova de Lisboa, Portugal)

## LOGIC FOR AI

"Implications in Vivid Logic," Seiki Akama and Hiroto Ohnishi (Fujitsu, Japan)

## METHODOLOGICAL ISSUES

"Viewpoints and Selective Inheritance in Object-Oriented Modeling," Markku Oivo (VTT, Finland)

"Linguistic Tools for Intelligent Systems," Boris Stilman (Univ. of Colorado, USA)

"'Competence-Switching' Managed by Intelligent Systems," Edeltraud Egger (Technical Univ. of Vienna, Austria) and Hardy Hanappi (Academy of Science, Austria)

"Utilization of the Case-Based Reasoning Method to Resolve Dynamic Problems," Sophie Rougegrez (Univ. Paris VI, France)

# IMPLICATIONS IN VIVID LOGIC

## Seiki Akama and Hiroto Ohnishi

Department of Information System, Teikyo University of Technology,
2289 Uruido, Ichihara-shi, Chiba, 290-01, JAPAN
Toyo Women's College
1660 Hiregasaki, Nagareyama-shi, Chiba, 270-01, JAPAN

## ABSTRACT

We discuss implications in vivid logic to enhance the expressive power concerning conditional knowledge. This can be accomplished as an extension of Wagner's partial logical framework. We show that the resulting system is equivalent to Nelson's constructive logic. We also argue inconsistency handling in the proposed framework.

## 1. INTRODUCTION

The idea of *vivid knowledge* was proposed by Levesque[1] to handle incomplete information in knowledge bases. However, to make Levesque's idea computationally feasible we have to incorporate both *closed world assumption*(CWA) and the *open world assumption*(OWA) in a unified framework. Unfortunately, as is well known these two meta-rules for handling negative information are incompatible.

The lesson from Levesque's observation is that there is a need to formalize two types of negation corresponding to CWA and OWA. In particular, we should deal with *explicit* negative information. For example, negation in logic programming has been considered as *negation as failure*(NAF) to avoid the computational overhead in implementing classical negation. But, recently several people have proposed to add explicit negation to logic programming effectively; see Gelfond and Lifschitz[2] and Kowalski and Sadri[3] for details. It is thus interesting to explore a logical framework for vivid reasoning.

Wagner[4][5] proposed a *vivid logic*(VL) to incorporate strong and weak negation within the framework of partial logic. In fact, vivid logic can be regarded as the first promising system for vivid reasoning. For instance, Wagner's system has some similarities with Nelson's[6] *constructive logic*. Constructive logic has attracted researchers in AI, e.g., Akama[7] and Pearce and Wagner[8]. The vivid logic can also be automated by means of logic programming with strong negation due to Pearce and Wagner[8].

The purpose of this paper is to expand Wagner's vivid logic with implication for describing conditional knowledge. By introducing implications in Wagner's logic, we can improve the expressive power required for vivid reasoning. From a theoretical point of view, the given system is also attractive in the sense that it corresponds to the full system of Nelson's constructive logic. This is because Nelson's

system can express both truth and falsity in a constructive setting.

The rest of this paper is structured as follows. In section 2, we review Wagner's vivid logic VL. Section 3 discusses Nelson's constructive logic. We proposed the extended vivid logic EVL in section 4. We develop a semantics and proof theory for EVL. We also try to embed EVL in an extended version of logic programming with strong negation. Section 5 argues the inconsistency handling in the proposed framework. Section 6 ends with our conclusions.

# 2. VIVID LOGIC

The language of Wagner's[4][5] vivid logic VL consists of &(conjunction), ∨(disjunction),-(weak negation), and true(truth). A literal is either an atomic formula or strongly negated atom. A vivid knowledge base V consists of inference rules of the form lit ← F read as lit if F, where lit is a literal and F is an arbitrary formula. We call such rules *conditional facts*. A rule with its premise true is called a *fact* denoted by lit instead of lit ← true. A variable-free expression is said to be *ground*.

Although we assume implicit quantification, a vivid knowledge base V with non-grounded conditional facts can be identified with a dynamic representation of the corresponding set of ground conditional facts by means of the current domain of individuals U, denoted by $[V]_U$:

$$[V]_U = \{lit\sigma \leftarrow F\sigma: lit \leftarrow F \in V \text{ and } \sigma: Var(lit, F) \rightarrow U\}$$

(2.1)

where σ ranges over all mappings from the set of variables of lit and F into the set of all constant symbols U. We call σ a ground substitution for lit ← F and $[V]_U$ the Herbrand expansion of V relative to a certain Herbrand universe U. We also denote by [V] the Herbrand expansion of V relative to the Herbrand Universe $U_V$.

Wagner argued that VL can represent four kinds of information, namely definite positive information, definite negative information, conditional information, and implicit negative information. The difference between strong and weak negation is that ¬p means that p is falsifiable and -p means that p is not verifiable, respectively.

We then describe a model theory for VL. Let $M = \langle M^+, M^- \rangle$ be a partial Herbrand interpretation, where $M^+$ contains positive facts and $M^-$ contains negative facts. A *model* of a program P is an interpretation satisfying all clauses of P. The partial Herbrand interpretation can define two forcing relations ⊨ and ⊨| to express provability and re futability, respectively:

$$M \models a \quad iff \quad a \in M^+$$

$$M \models F \& G \quad iff \quad M \models F \text{ and } M \models G,$$

$$M \models F \vee G \quad iff \quad M \models F \text{ or } M \models G,$$

$$M \models \neg F \quad iff \quad M \models| F,$$

$$M \models -F \quad iff \quad M \not\models F,$$

$$M \dashv a \quad iff \quad a \subseteq M^{-},$$

$$M \dashv F \ \& \ G \quad iff \quad M \dashv F \quad or \quad M \dashv G,$$

$$M \dashv F \lor G \quad iff \quad M \dashv F \quad and \quad M \dashv G,$$

$$M \dashv \neg F \quad iff \quad M \vDash F,$$

$$M \dashv -F \quad iff \quad M \nvDash F.$$

In this model, we assume that $M \vDash$ true for all models $M$. $M$ is a *partial Herbrand model* of V, denoted by $M \vDash V$, if for all lit $\leftarrow F \in [V]$ and any ground instance $F\sigma$ of F, $M \vDash F\sigma$ implies $M \vDash$ lit$\sigma$. We say that F is a *logical consequence* of P, denoted by $P \vDash F$, if every model of P is also a model of F. We call $M'$ an *extension* of $M$, denoted by $M' \geq M$, if $M^{+} \subseteq M'^{+}$ and $M^{-} \subseteq M'^{-}$.

*Theorem 2.2* (Wagner[4])
Let $M' \geq M$ and let F be a ground formula without weak negation, then

$$M \vDash F \implies M' \vDash F,$$

$$M \dashv F \implies M' \dashv F.$$

*Theorem 2.3* (Wagner[4])
Every vivid knowledge base V without weak negation has a least model.

A proof theory for VL can be developed by formalizing a derivability relation using natural deduction. We use the notation $V \vdash F$ to show that F is derivable from V:

$$(\&) \qquad V \vdash F \quad and \quad V \vdash G \implies V \vdash F \ \& \ G,$$

$$(\neg\&) \qquad V \vdash \neg F \quad or \quad V \vdash \neg G \implies V \vdash \neg(F \ \& \ G),$$

$$(-\&) \qquad V \vdash -F \quad or \ V \vdash -G \implies V \vdash -(F \ \& \ G),$$

$$(-\neg\&) \qquad V \vdash -\neg F \quad and \quad V \vdash -\neg G \implies V \vdash -\neg(F \ \& \ G),$$

$$(\neg\neg) \qquad V \vdash F \implies V \vdash \neg\neg F,$$

$$(-\neg\neg) \qquad V \vdash -F \implies V \vdash -\neg\neg F,$$

$$(\neg-) \qquad V \vdash F \implies V \vdash \neg-F,$$

$$(--) \qquad V \vdash F \implies V \vdash --F,$$

$$(-\neg-) \qquad V \vdash -F \implies V \vdash -\neg-F,$$

where F and G are ground formulas. We assume that for any V, $V \vdash$ true. Let V be a set of simple facts. Then, the derivability of a fact from V can be defined as a membership check in the following way:

$$(lit) \qquad V \vdash lit \quad iff \quad lit \in V.$$

$$(-lit) \qquad V \vdash -lit \quad iff \quad lit \notin V.$$

In a general case, we can define the derivability as

$$(lit) \qquad V \vdash lit \quad iff \quad \exists (lit \leftarrow F) \in [V]: V \vdash F,$$

$$(-lit) \qquad V \vdash -lit \quad iff \quad \forall (lit \leftarrow F) \in [V]: V \vdash -F.$$

However, this definition only works for well-founded VKB. For a general case, we need to add the following deduction rules:

(2.4)    V - F for some lit ← F ∈ [V]  ==>  V - lit,

         V - -F for ... lit ← F ∈ [V]  ==>  V - -lit.

A formula F is thus derivable from V if there is a derivation for the sequent V - F. Wagner also discussed weakly well-founded VKB and a loop-tolerant recursive proof theory.

# 3. CONSTRUCTIVE LOGIC

Nelson[6] proposed constructive logic to formalize the notion of *constructible falsity* or *strong negation* to overcome the weakness of intuitionistic negation. In this sense, we distinguish constructive logic from Heyting's intuitionistic logic. Nelson's principal aim is to describe negation in the same way as in constructive truth. Then we need strong negation satisfying some of the classical principles. For constructive logic, the reader is referred to Gabbay[9] and Akama[10][11][12][7][13] for details.

The axiomatization of constructive logic with strong negation denoted by N can be given as that of positive intuitionistic predicate logic with the following axioms for strong negation:

(A1)   $-A \rightarrow (A \rightarrow B)$,

(A2)   $-(A \& B) \leftrightarrow (-A \lor -B)$,

(A3)   $-(A \lor B) \leftrightarrow (-A \& -B)$,

(A4)   $-(A \rightarrow B) \leftrightarrow (A \& -B)$,

(A5)   $--A \leftrightarrow A$,

(A6)   $-\forall xA(x) \leftrightarrow \exists x-A(x)$,

(A7)   $-\exists xA(x) \leftrightarrow \forall x-A(x)$,

being closed under *detachment* and two obvious quantificational rules. If we delete (A1) from N, the resulting system is said to be $N^-$.

Intuitionistic negation "¬" can be defined in N as one of the following forms; see Nelson[6]:

$\neg A = A \rightarrow B \& -B$                    (3.1)

$\neg A = A \rightarrow -A$                          (3.2)

We also have the following as a theorem in N:

$-A \rightarrow \neg A$                              (3.3)

For a semantics for N, we can give a Kripke type model theory by extending intuitionistic Kripke semantics. Following Gabbay[9], let (S,R,0,val) be a *strong propositional Kripke structure*, where (S,R,0) is a partially ordered set with first element $0 \in S$ and val is a three-valued function such that for each $t \in S$ and atomic q, $val(t,q) \in \{-1, 0, 1\}$ satisfying that tRs and $val(t,q) \neq 0$ imply $val(t,q) = val(s,q)$. Intuitively, the values of val 1, 0, -1 are, respectively, used to express truth, undefined, falsity in a constructive sense.

The truth-value $(A)_t$ of a formula A at a point t of a strong

propositional Kripke structure is defined by induction as:

$(A)_t = val(t,A)$ for atomic A,

$(A \& B)_t = min((A)_t, (B)_t)$,

$(A \lor B)_t = max((A)_t, (B)_t)$,

$(A \to B)_t = 1$ iff for all s, tRs and $(A)_s=1$ imply $(B)_s=1$,

$(A \to B)_t = -1$ iff $(A)_t = 1$ and $(B)_t = -1$,

$(\neg A)_t = 1$ iff $(A)_t = -1$.

We say that A is *valid* in the structure if $(A)_0 = 1$. We can establish the completeness of N as follows:

*Theorem 3.4* (Completeness theorem)

$\vdash_N A$ iff A is valid in every strong propositional Kripke structure.

The above Kripke semantics can also be extended for the predicate logic; see Thomason[14] and Akama[11][13].

# 4. AN EXTENSION OF VIVID LOGIC WITH IMPLICATION

We are now in a position to extend Wagner's vivid logic VL with intuitionistic implication to formalize vivid reasoning within the framework of constructive logic. It is obviously of importance to add implications to VL because we can express conditional knowledge flexibly.

We introduce (intuitionistic) implication $\to$ to VL. The resulting system is called extended vivid logic EVL. Then we can use embedded implications to describe conditional knowledge in connection with hypothetical reasoning. In EVL, implications are interpreted in the following way:

$M \models F \to G$ iff $\forall M' \supseteq M$ (M' $\models$ A implies M' $\models$ B),

$M \dashv F \to G$ iff $M \models F$ and $M \dashv G$.

The derivability relation of the implication can be expressed as

$(\to)$  V, F $\vdash$ G ==> V $\vdash$ F $\to$ G,

$(\neg\to)$ V $\vdash$ F and V $\vdash$ $\neg$G ==> V $\vdash$ $\neg$(F $\to$ G).

Next, we prove the relationship of EVL and Nelson's constructive logic by showing the equivalence between partial Herbrand semantics and Kripke semantics. The proof can be developed using a variant of strong Kripke structure defined above in such a way that Herbrand interpretations are defined as subsets of the Herbrand base. Let a *Kripke interpretation* M for A be a tuple $\langle W, \subseteq, V_+, V_-, D \rangle$, where $W \subseteq 2^{B(A)}$ is a partially ordered set of worlds, $V_+$ and $V_-$ are functions which map every literal to a subset of W closed under $\subseteq$ satisfying $V_+(a) \cap V_-(a) = \emptyset$, and D is a (constant) domain. We here denote the Herbrand base for P by B(P). Then we can define two forcing relations $\models$ and $\dashv$ with respect to a formula A in a world w of M, denoted by M, w $\models$ A (M, w $\dashv$ A) to state that A is true (false) at a world w in a

model $M$ as follows:

$$M, w \models a \quad \text{iff} \quad w \in V_+(a),$$

$$M, w \dashv a \quad \text{iff} \quad w \in V_-(a).$$

The forcing relations can thus be extended by induction on A as

$$M, w \models \text{true},$$

$$M, w \dashv \text{false},$$

$$M, w \models F \ \& \ G \quad \text{iff} \quad M, w \models F \text{ and } M, w \models G,$$

$$M, w \models \neg F \quad \text{iff} \quad M, w \dashv F,$$

$$M, w \models -F \quad \text{iff} \quad M, w \not\models F,$$

$$M, w \models F \rightarrow G \quad \text{iff} \quad \forall w' \supseteq w \ (M, w' \models F \Longrightarrow M, w' \models G),$$

$$M \models F(x) \quad \text{iff} \quad M \models F(t) \text{ for some } t \in D,$$

$$M, w \dashv F \ \& \ G \quad \text{iff} \quad M, w \dashv F \text{ or } M, w \dashv G,$$

$$M, w \dashv \neg F \quad \text{iff} \quad M, w \models F,$$

$$M, w \dashv -F \quad \text{iff} \quad M, w \not\models F,$$

$$M, w \dashv F \rightarrow G \quad \text{iff} \quad M, w \models F \text{ and } M, w \dashv G,$$

$$M \dashv F(x) \quad \text{iff} \quad M \models F(t) \text{ for all } t \in D.$$

We say that a Kripke interpretation satisfies a formula A iff $M, w_0 \models$ A, where $w_0$ is the least (actual) world. A goal G is a logical consequence of the knowledge base V, denoted by $V \models_K G$ iff $M, w_0 \models V \Longrightarrow M, w_0 \models G$ for all Kripke interpretations M for P. We here write PB(A) for a partial Herbrand base for A defined as a pair $\langle B(A), \{\neg a : a \in B(A)\}\rangle$.

*Lemma 4.1*

Let $I_0$ be a subset of PB(A) for a formula A and M the Kripke interpretation $\langle W, \subseteq, V_+, V_-, D \rangle$. If $W = \{I' : I' \in 2^{PB(A)}$ and $I_0 \subseteq I'\}$, where $I_0$ is a subset of PB(A), then for all $I \in W$,

$$I \models A \quad \text{iff} \quad M, I \models A.$$

Proof: By induction on A.

*Lemma 4.2*

$$I \models G \quad \Longleftrightarrow \quad M, I \models G.$$

Proof: By induction on G.

*Theorem 4.3*

For any vivid knowledge base V and any goal G,

$$V \models_K G \quad \text{iff} \quad V \models G.$$

Proof: For (=>), we have $M, I_0 \models P \Longrightarrow M, I_0 \models G$ for all Kripke interpretations by hypothesis. Let I be an interpretation satisfying $I \models P$. Then, $M, I \models P$ by lemma 4.1, where M and I are as described above. So, $M, I \models G$ by hypothesis. From lemma 4.2, we obtain $I \models G$.

To prove (<=), by hypothesis $I \models P \Longrightarrow I \models G$ for all interpretations. Then, we must prove that $M, I_0 \models P \Longrightarrow M, I_0 \models G$ for all Kripke interpretations. There are two cases for $I_0$. In other words, either $I_0 \not\models P$ or $I_0 \models G$. If $I_0 \not\models P$ then $M, I_0 \not\models P$ by lemma 4.2. Similarly, if $I_0 \models G$ then $M, I \models_0 G$ by lemma 4.2. As a consequence, we have either $M, I_0 \not\models P$ or $M, I_0 \models G$. This implies that $M, I_0 \models P \Longrightarrow M, I_0 \models G$.

Because Pearce and Wagner's[8] *logic programming with strong negation*(LPS) is faithful to Nelson's logic N⁻, EVL without "not" is also faithful to not only N⁻ but also LPS with intuitionistic implications. This implies that EVL can be simulated within LPS. In other words, LPS can serve as a general constructive framework for implementing vivid reasoning.

# 5. WEAK NEGATION AND INCONSISTENCY

In this section, we discuss the issue of weak negation in relation to reasoning about inconsistency. First, we describe how negation as inconsistency can be interpreted in EVL. In fact, EVL has weak negation "not" as negation by failure, which is, however, meta-level negation. To make EVL more logical, we would like to define weak negation as in constructive logic, as intuitionistic negation can be defined in N. Thus, we introduce weak negation in EVL identifying with intuitionistic negation in the manner of negation as inconsistency.

*Negation as inconsistency*(NAI) was proposed by Gabbay and Sergot [15] as an alternative to NAF in order to give a logical negation to logic programming. NAI can express negation by means of inconsistency checking. In fact, Gabbay and Sergot's formulation assumes that a database can be identified with a positive program P augmented with a set of negative clauses N of the form of integrity constraints. If both positive goal G and negative goal ¬G are proved from the database, then NAI denoted by "not*" can be confirmed in the sense that:

$$(P,N)?not^* A = 1 \quad iff \quad (P,A)?B = 1 \qquad (5.1)$$

where $B \in N$. Unfortunately, NAI is not local negation since it is based on indirect interpretation via inconsistency checking. Thus NAI cannot express explicit negative information. This means that NAI is a kind of weak negation.

Gabbay and Sergot showed that NAF is a special case of NAI by assuming N as a set of atoms which finitely fail from P, namely

$$P(?F)G = 1 \quad iff \quad (P,N)(?I)G = 1 \qquad (5.2)$$

where $P(?F)G = 1$ $((P,N)(?I)G = 1)$ denotes success computation of G from P ((P,N)) by NAF (NAI). Gabbay and Sergot also showed the translation of NAI into N-Prolog of Gabbay and Reyle[16]:

*Definition 5.3*
Let (P,N) be a database, and let "false" be a symbol for absurdity:
(a) Define the following translation * from the language of NAI into N-Prolog:

   (a1) $q^* = q$ for atomic,

   (a2) $(A \& B)^* = A^* \& B^*$,

   (a3) $(A \rightarrow B)^* = A^* \rightarrow B^*$,

   (a4) $(not^* A)^* = A^* \rightarrow$ false.

(b) Let $D^*$ be the following database of N-Prolog:

$$D^* = \{A^*: A \in P\} \cup \{B^* \rightarrow \text{false}: B \in N\}.$$

$D^*$ is called the translation of $(P,N)$.

*Theorem 5.4*
$(P,N)(?I)G = 1$  iff $D^*?G^* = 1$ in N-Prolog.

This result can be recast in our case by formalizing N-Prolog with strong negation extending Gabbay and Reyle's language. As N-Prolog is in fact complete for intuitionistic logic, $A \rightarrow$ false is identified with intuitionistic negation $-A$. This interpretation is in parallel with the definition of intuitionistic negation in N, namely

$$-A = A \rightarrow \neg A \tag{5.5}$$

By (5.5), we modify the translation of "not*" as follows:

(a4')  $(\text{not}^* A)^* = A^* \rightarrow (\neg A)^*$,

where $(\neg A)^* = \neg A^*$ for atomic A. We should also rewrite (b) as

(b')  $D^* = \{A^*: A \in P\} \{B^* \rightarrow (\neg B)^*: B \in N\}$

It is easy to understand that, NAI (also NAF) can be expressed as intuitionistic negation. Therefore, NAI is also complete for Nelson's constructive logic N and EVL. Unfortunately, this result cannot be applied to the case where underlying logic is replaced by the weaker system $N^-$. This is because $N^-$ is a *paraconsistent logic*, which is a logical system in which inconsistency does not mean triviality; see Priest, Routley and Norman[17].

Second, we discuss the problem of inconsistency resolution. One way to overcome difficulties with inconsistency is to employ paraconsistent logic which can view a contradiction true. Namely, when A & $\neg A$ is true, we can both assume that both A and $\neg A$ are true. Unfortunately, the paraconsistent approach cannot extract reasonable information from the inconsistent database. Since EVL can be considered as a version cf paraconsistent logic, similar difficulty arises. Thus, we must work out a method of resolving inconsistency in our system.

As paraconsistent constructive logic $N^-$ (which is also a basis for EVL), can tolerate inconsistency, we can utilize the underlying system for conflict resolution. This is because $N^-$ can still do constructive reasoning under inconsistency. Our architecture thus deduces contradictions by means of $N^-$, and to resolve it so that we can get a plausible conclusion based on some meta-level control strategy. However, it is very difficult to carry out the idea in EVL mainly due to the presence of weak negation. Consider the following example:

*Example 1.*
Let VKB be the following:

fly(x) ← bird(x), -abnormal(x)

¬fly(tweety)

bird(tweety)

Existing logic programming semantics cannot capture the intended meaning of the VKB. In fact, there is no model in well-founded semantics

due to inconsistency. We can thus prove both fly(tweety) and ¬fly(tweety). However, the VKB intends to express that Tweety does not fly explicitly. In this sense, we should deduce a plausible conclusion ¬fly(tweety) even if inconsistency arises.

One possible solution is to amalgamate general rules and exceptions in the sense of Kowalski and Sadri's[3] *logic programs with exceptions.* In their system, the negative conclusion ¬A is adopted if we conclude both A and ¬A, simultaneously. It is, however, rather ad hoc, since the postulate is definitely false in common-sense reasoning. The problem with the example is that the *closed world assumption*(CWA) by way of NAF is unrestrictedly applied. Since CWA is expressed as the rule ¬A ← -A in Gelfond and Lifschitz's[18][2] answer set semantics, we obviously face the defect. The lesson from the fact is that in logic programming with two kinds of negation we have to flexibly tell closed world reasoning (i.e. weak negation) from open world reasoning (i.e. explicit negation).

This suggests that if contradiction appears and if one of the complementary conclusions was derived by CWA, then we can resolve inconsistency by discarding the conclusions induced by CWA. In other words, we need an action to revise VKB so that we can block the application of CWA in relation to the inconsistency. Using this strategy, we get the revised VKB

(5.6)     fly(x) ← bird(x), -abnormal(x)

        ¬fly(tweety)

        bird(tweety)

        abnormal(x) ← -abnormal(x)

We can then prove the plausible conclusion ¬fly(tweety). The point is to add the fourth clause to avoid the application of CWA. We can thus see that NAF is not logically sound in vivid reasoning. The above strategy can easily be described by means of meta-level:

      find_naf_body(V,A,-B) ← prove (V,A), prove(V,¬A)

      prove(V",¬A) ← cwa_prove(V,-B,A), C = (B ← -B),

                          add(V,C,V'), delete(V',¬A,V")

      prove(V",A) ← cwa_prove(V,-B,¬A), C = (B ← -B),

                          add(V,C,V'), delete(V',¬A,V")

where "find_naf_body(V,A,-B)" is to mean that if contradiction A & ¬A is provable, find weak negation in a body, "cwa_prove(V,-B,A)" that A is provable from V by CWA to assume -B, "add(V,A,V')" that V' is obtainable from V by adding A, and "delete(V,A,V')" that V' is obtainable from V by deleting A, respectively. The meta-program is a natural specification of our strategy. It is, however, open whether the strategy can be established in the object-level computation.

The next example is called *Barber's Paradox*, which is of interest to the formalization of common-sense reasoning.

*Example 2.*

Let VKB be given as follows:

shave(fred, x) ← -shave(x,x)

The problem is: does the barber Fred shave himself? The well-founded semantics can conclude that he does not. However, VKB proves that shave(fred,fred). The reason is that we have -shave(fred,fred) since shave(fred,fred) is not contained in VKB. As a consequence, we can deduce that shave(fred,fred). Next, we add "shave(casanova,casanova)" to VKB. Then we can prove both shave(fred, bill) and -shave(fred, casanova). Consider the case that we instead add "mayor(casanova)". In this case, no semantics including our method cannot conclude that Fred shaves Casanova. But, from the view point of common-sense reasoning, we would like to deduce shave(fred, casanova). This example suggests that we can incorporate some empirical evidence into reasoning about incomplete information. In other words, vivid logic also has to employ meta-level reasoning in addition to inference engine based on constructive logic.

# 6. CONCLUSIONS

We have introduced intuitionistic implication into Wagner's vivid logic. We have also shown how NAI(NAF) can be interpreted in our logic. It has been argued that EVL with its device to resolve contradictions is a promising theory for reasoning about inconsistency.

# REFERENCES

1.   Levesque,H.(1986): Making believers out of computers, *Artificial Intelligence 30*, 81-107.
2.   Gelfond,M. and Lifschitz,V. (1991): Logic programs with classical negation and disjunctive databases, *New Generation Computing 9*, 365-385.
3.   Kowalski,R.A. and Sadri,F. (1990): Logic programs with exceptions, D...D.Warren and P. Szeredi(eds.), *Proc. of the 7th ICLP*, 598-613, MIT Press, Cambridge.
4.   Wagner,G.(1991a): A database needs two kinds of negation, B. Thalheim, J.Demetrovics and H.-D.Gerhardt(eds.), *Proc. of the 3rd Symposium on Mathematical Foundations of Database and Knowledge Base Systems*, 357-371, Springer, Berlin.
5.   Wagner,G.(1991b): Logic programming with strong negation and inexact predicates, *Journal of Logic and Computation 1*, 835-859.
6.   Nelson,D.(1949): Constructible falsity, *The Journal of Symbolic Logic 14*, 14-26.
7.   Akama,S. (1989): *Constructive Falsity: Foundations and Their Applications to Computer Science*, Ph.D. dissertation, Department of Administration Engineering, Keio University, Yokohama, Japan.
8.   Pearce,D. and Wagner,G.(1991): Logic programming with strong negation, P.Schroeder-Heister(ed.), *Proc. of Workshop on Extensions in Logic Programming*, 311-376, Springer, Berlin, 1991.
9.   Gabbay,D. (1981): *Semantical Investigations in Heyting's Intui-*

*tionistic Logic*, Reidel, Dordrecht.

10.  Akama,S. (1987): Resolution in constructivism, *Logique et Analyse 120*, 385-399.

11.  Akama,S. (1988a): Constructive predicate logic with strong negation and model theory, *Notre Dame Journal of Formal Logic 29*. 18-27.

12.  Akama,S. (1988b): On the proof method for constructive falsity, *Zeitschrift fur mathematische Logik und Grundlagen der Mathematik 34*, 385-392.

13.  Akama,S. (1990): Subformula semantics for strong negation systems, *Journal of Philosophical Logic 19*, 217-226.

14.  Thomason,R.H. (1969): A semantical study of constructible falsity, *Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik 15*, 247-257.

15.  Gabbay,D. and Sergot,M. (1986): Negation as inconsistency I, *Journal of Logic Programming 3*, 1-35.

16.  Gabbay,D. and Reyle,U.(1984): N-Prolog: An extension of Prolog with hypothetical implication I, *Journal of Logic Programming 1*, 319-355.

17.  Priest,G., Routley,G. and Norman,J. (1989): *Paraconsistent Logic: Essays on the Inconsistent*, Philosophia Verlag, Munchen.

18.  Gelfond,M and Lifschitz,V.(1990): Logic programs with classical negation, D.H.D.Warren and P. Szeredi(eds.), *Proc. of the 7th ICLP*, 579-597.

# A SELF-LEARNING BAYESIAN EXPERT SYSTEM (B.E.ST.)

Farrokh Alemi*, Pallav Bhatt*, Eric Eisenstein*,
Adam Fadlalla**, Richard Stephens***, and John Butts*

*Health Administration Program
**Computer and Information Science Department
***Department of Sociology
Cleveland State University
Cleveland, OH 44115

## ABSTRACT

This paper presents a Bayesian Expert System (B.E.St.). This system is different from other forms of supervised machine learning in that it induces likelihood ratios from a set of cases. B.E.St. provides unique methods for managing conditional dependencies, for selecting questions to use in its models, and for predicting events when given responses that are not contained in its original set of cases.

## 1. TERMINOLOGY

B.E.St. predicts the probability that one of two mutually exclusive and collective exhaustive target events will occur. We identify a target event as H, indicate the presence of the event as $H_1$, and indicate the absence of the event as $H_0$. Probability is represented by the small letter p. Thus, the probability of the event $H_1$ may be written as $p(H_1)$.

B.E.St. assumes that answers to a set of questions developed by experts can serve as clues for predicting the target event. In this paper, the terms "answers," "prediction clues," and "data" are used interchangeably, and they are identified as $D_1$, $D_2$, ..., $D_n$. Thus, $D_i$ represents the answer to the ith question, the ith datum, and the ith clue in the prediction task.

The possibility of the target event can be written as the conditional probability of $H_1$ given the various clues:

$$p(H_1|D_1, D_2, ..., D_n).$$

The posterior odds for the occurrence of the target event is shown as:

Posterior Odds of H = $p(H_1|D_1, D_2, ..., D_n)/p(H_0|D_1, D_2, ..., D_n)$

B.E.St. is designed to calculate the posterior odds of a specified target event. These odds are calculated as a function of likelihood ratios. The likelihood ratio of $D_i$ is defined as:

Likelihood Ratio of $D_i$ = $p(D_i|H_1)/p(D_i|H_0)$

Where $p(D_i|H_1)$ is the probability of observing clue $D_i$ in cases where $H_1$ has occurred and $p(D_i|H_0)$ is the probability of observing the same clue in cases where $H_0$ has occurred. For example, if we are examining the importance of shock in predicting in-hospital mortality, then the likelihood ratio principle suggests that this importance is equal to the prevalence of shock among patients who live divided by the prevalence of shock among patients who die:

$$\text{Likelihood Ratio of Shock} = p(\text{Shock}|\text{Live})/p(\text{Shock}|\text{Die})$$

A likelihood ratio is a non-negative score. Values larger than one suggest qualitatively greater support for the occurrence of the target event, while unity suggests that no evidence exists for or against the target event.

## 2. KNOWLEDGE MANAGEMENT APPROACH

There are numerous approaches to reasoning. Many expert systems incorporate heuristics or rules provided by experts in a particular field. Such an approach is advantageous because it is flexible and easily understood by experts. However, this approach has two disadvantages: (1) it requires a substantial time commitment on the part of the experts and (2) it fails to provide a clear mechanism for resolving contradictions in experts' opinions. Other supervised learning systems, such as neural networks, do not require that experts provide rules as these systems assess the relationships between clues and the target event through automatic analysis of data and goodness of fit principles. However, these systems often require that all contraditions in their data be resolved before a solution can be determined.

## 2.1 BAYESIAN REASONING

B.E.St. has many advantages of both heuristic-based expert systems and neural networks. B.E.St. is flexible and takes little time to set up, and, unlike a neural network, its inference mechanism can be easily understood by experts. Furthermore, unlike heuristic-based expert systems, B.E.St. is capable of resolving the contradictory opinions of one or more experts.

B.E.St. predicts the target event using probability rules. If $p(D_1, D_2, \ldots, D_n|H_1)/p(D_1, D_2, \ldots, D_n)$ describes the likelihood ratio associated with clues $D_1$ through $D_n$, then B.E.St. calculates the odds for the target event using a formula first suggested by Bayes:[1]

$$\text{Odds of } (H|D_1, D_2, \ldots, D_n) = $$
$$[p(D_1, D_2, \ldots, D_n|H_1)/p(D_1, D_2, \ldots, D_n|H_0)]*[p(H_1)/p(H_0)] \qquad (1)$$

---

1 T. Bayes, "Essays toward solving a problem in the doctrine of changes," *Philosophical Transaction of Royal Society*, 53: 370-418 (1783).

The principles underlying Bayes' formula include the use of likelihood ratios. Although controversy exists regarding the use of Bayes' formula for subjective inferences, the use of likelihood ratios is well accepted by statisticians.[2]

## 2.2 BAYESIAN LEARNING

Machine learning occurs in at least two different ways: (1) through deduction from a set of rules or (2) through induction from a set of cases.[3] Our focus here is on inductive learning. Researchers who study automated learning suggest several methods for inductive learning. In one approach to supervised learning, a case example is used to generate an 'if-then' rule.[4] Subsequent counter-examples are used to refine the rule with a series of 'unless' conditions. The process continues until a set of 'if-then' and 'unless' rules can explain all cases. Another approach to unsupervised learning searches for regularities among cases by clustering similar cases together.[5]

While the above methods of automated learning are reasonable in many domains, we found that inducing likelihood ratios from a set of cases is another valid method of machine learning. For some time, statisticians have argued that the only way to properly measure the impact of a clue in an inference task is with the likelihood ratio.[6] Figure 1 describes the system's knowledge base and the method it uses to calculate the likelihood ratio associated with a clue.

The system's knowledge base contains previous users' answers and subsequent observations related to the occurrence of the target event. The knowledge base for m previous users who have answered n questions is as shown in Figure 1. Entries under the first column are the number of cases in the knowledge base. Entries under the question columns are the answers to questions that were provided by previous users. For example,

---

2  A. W. F. Edwards, "The History of Likelihood," *International Statistical Review*, 42: 8-15 (1974).

3  R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine Learning*, Morgan Kaufmann Publishers, Inc., Los Altos, California (1986).

4  P. H. Winston, "Learning by Augmenting Rules and Accumulating Censors", *Machine Learning: An Artificial Intelligence Approach, Vol II*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (eds.), Morgan Kaufmann Publishers, Inc., Los Altos, California (1986).

5  R. E. Stepp III and R. S. Michalski, "Conceptual Clustering: Inventing Goal-Oriented Classifications of Structured Objects", *Machine Learning: An Artificial Intelligence Approach, Vol. II*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (eds.), Morgan Kaufmann Publishers, Inc., Los Altos, California (1986).

6  R. A. Fisher, *Statistical Methods and Scientific Inference*, (2nd ed rev. 1959), Hafner, London, Oliver and Boyd, New York (1956)

if the first entry for question 1 in column 2 is a 2, this means that the first user, case number 1, answered option 2 for question 1. An entry of -1 means that an answer is missing from the data set. The entry under the hypothesis column indicates whether the target event actually occurred. Specifically, a value of 1 means that the target event has occurred, while a value of 0 means that the target event has not occurred. No missing values are allowed in the hypothesis column. The knowledge base may be specified by experts or may be established by collecting data on actual events. B.E.St. utilizes the information in its knowledge base to calculate the likelihood ratios associated with each clue.

Figure 1. System knowledge base organization

| Case | Question 1 | Question 2 | . . . . . | Question n | Hypothesis |
|------|-----------|-----------|-----------|-----------|-----------|
| 1 | 2 | 2 | . . . . . | -1 | 1 |
| 2 | 9 | 1 | . . . . . | 1 | 0 |
| 3 | 2 | 2 | . . . . . | 1 | 1 |
| . . . | . . . | . . . | . . . . . | . . . | . . . |
| m | 1 | 1 | . . . . . | -1 | 1 |

## 3. MANAGING CONDITIONAL DEPENDENCE

### 3.1 CONDITIONAL LIKELIHOOD RATIOS

B.E.St. assesses likelihood ratios while accounting for interdependencies that may exist among clues. There are a number of mathematical methods for accounting for interdependencies. These methods include various equivalent probability formulas, the use of dependence trees, and the use of correlations among pairs of clues.[7, 8, 9, 10, 11, 12]

7   D. J. Croft, Is Computer Diagnosis Possible, *Computers and Biomedical Research*, 5:   351-367 (1972).

8   M. J. Norusis and J. A. Jacquez, Diagnosis.   I.   Symptom Nonindependence in Mathematical Models for Diagnosis, *Computers and Biomedical Research*, 8:   156-172 (1975).

9   B. Seroussi, ARC and AURC Cooperative Group, Computer Aided Diagnosis of Acute Abdominal Pain When Taking Into Account Interactions," *Methods of Information in Medicine*, 25:   194-198 (1986).

10   C. Ohmann, Q. Yang, M. Kunneke, H. Stolzing, K. Thon and W. Lorenz, "Bayes Theorem and Conditional Dependence of Symptoms:   Different Models Applied to Data of Upper Gastrointestinal Bleeding," *Methods of*

In addition, there are behavioral approaches that may be used to account for interdependencies, whereby experts are asked to group clues into independent clusters.[13,14] We chose a mathematical approach to account for the interdependence of clues, so that modifications could be calculated automatically by the computer. B.E.St. uses the following probability rule to assess the joint likelihood ratio of a set of dependent clues:

$$p(D_1, D_2, \ldots, D_n|H_1) =$$
$$p(D_1|H_1)*p(D_2|H_1, D_1)*p(D_3|H_1, D_1, D_2)*$$
$$p(D_4|H_1, D_1, D_2, D_3)*\ldots*p(D_n|H_1, D_1, D_2, D_3, \ldots,D_{n-1}) \qquad (2)$$

Note that each term in the above formula is conditioned on answer to previously asked questions. The first term is conditioned on no answers since no questions were previously asked; the second term is conditioned on answers to the first question; the third term is conditioned on answers to the first two questions; and the last term is conditioned on answers to the first n-1 questions. B.E.St. follows the above rule, and every time an answer is given, it calculates all subsequent likelihood ratios conditioned on the user's previous answers.

To calculate conditional likelihood ratios, the system automatically reduces the data base to cases in which the condition is met. For example, if the answer to the first question is that the patient is in shock, then all cases of non-shock patients are not incorporated in the calculations of likelihood ratios for subsequent answers. Since likelihood ratios are recalculated dynamically, this approach allows the interdependence among the clues to change. Thus, two clues that are dependent when the full data set is analyzed may be independent when the reduced data set is used and vice versa. This dynamic approach is more sensitive than an approach that calculates the interdependence of clues at the beginning of the data analysis.

---

*Information in Medicine*, 27: 73-83 (1988).

11  S. Lichtenstein, "Conditional Non-Independence of Data in a Practical Bayesian Task," *Organizational Behavior and Human Performance*, 8: 21-25 (1972).

12  A. Gammerman and A. R. Thatcher, "Bayesian Diagnostic Probabilities Without Assuming Independence of Symptoms," *Methods of Information in Medicine*, 30: 187-193 (1991).

13  H. J. B. Moens and J, K. Van Der Korst, "Comparison of Rheumatological Diagnoses by a Program and by a Physician," *Methods of Information in Medicine*, 30: 187-193 (1991).

14  D. H. Gustafson, J. J. Kestly, R. L. Ludke, and F. Larson, "Probabilistic Information Processing: Implementation and Evaluation of a Semi-PIP Diagnostic System," *Computers and Biomedical Research*, 6: 355-370 (1973).

The primary disadvantage of this dynamic approach, and of approaches for accounting for interdependencies in general, is that large data bases are required. As the number of conditions increases, more and more data become irrelevant and the size of the knowledge base decreases. For example, if a user indicates that the patient is in shock, B.E.St. ignores all of the data collected for non-shock patients. If shock and non-shock patients are equally probable, the data base is reduced by one-half. Likewise, each subsequent equally probable question can reduce the data base by one-half. Asking 8 equally probable questions will reduce the data by $2^8$, or 256 times. Therefore, the application of Formula 2 requires large knowledge bases. In order to overcome this difficulty, we allow approximation of Formula 2, as discussed in the following section.

## 3.2 APPROXIMATING CONDITIONAL DEPENDENCIES

A number of authors have suggested methods that could account for some of the dependencies among the clues. A method used in most statistical approaches, e.g. discriminant analysis or logistic regression, relies upon the correlations among the clues. Another approach, first suggested by Lincoln and Parker, allows users to condition each answer on the answer to one previous question.[15] B.E.St. expands the latter approach to allow users to specify the number of previous answers on which the calculation of subsequent likelihood ratios should be conditioned; the answers chosen for inclusion in subsequent calculations are those that are the most predictive. If this number is zero, then the system assumes that all answers are conditionally independent, i.e.:

$$p(p(D_1, D_2, \ldots, D_n|H_1) =$$
$$p(D_1|H_1)*p(D_2|H_1)**p(D_3|H_1)*p(D_4|H_1)*\ldots*pD_n|H_1) \quad (3)$$

If the number is n-1, then all answers are assumed to be dependent and equation 2 is used. If the number is between zero and n-1, some dependencies are accounted for while other are ignored. For example, if the user specifies the number 2, the likelihood ratios should be conditioned on no more than two previous answers. B.E.St. will automatically choose the two most predictive answers as defined below:

$$p(D_1, D_2, \ldots, D_n|H_1) = p(D_1|H_1)*p(D_2|H_1, D_1)*$$
$$p(D_3|H_1, D_1, D_2)*p(D_4|H_1, D_i, D_j)*\ldots*p(D_n|H_1, D_k, D_1) \quad (4)$$

Where i and j are chosen so that the difference between $p(D_4|H_1, D_i, D_j)$ and $p(D_4|H_1, D_1, D_2, D_3)$ is minimized and k and l are chosen so that the difference between $p(D_n|H_1, D_k, D_1)$ and $p(D_n|H_1, D_1, D_2, D_3, \ldots, D_{n-1})$ are minimized. In this manner, most of the dependencies are accounted for and more effective usage is made of a limited amount of learning data.

---

15  T. L. Lincoln and R. D. Parker, "Medical Diagnosis Using Bayes Theorem", *Health Services Research*, 2:34 (1967).

## 4. QUESTION SELECTION

### 4.1 TEST FOR STATISTICAL SIGNIFICANCE

Researchers have for some time known that the performance of a self-learning expert system depends upon the number of cases in its knowledge base. Through simulations, some researchers have found that the number of cases should be greater than 200 and less than 500, provided that the questions are conditionally independent.[16] An alternative approach for determining the appropriate size for a knowledge base is to rely on statistical sample theory.

In B.E.St. every time a question is answered, the knowledge base is reduced, which in turn changes the likelihood ratios associated with subsequent answers. If there are too few cases in the reduced knowledge base, then the calculated likelihood ratios may reflect random error and thus, may be misleading. Random variations suggest that a likelihood ratio is close to one and that the answer associated with the ratio has little or no statistical significance. B.E.St. was designed to examine whether changes in the knowledge base have led to likelihood ratios that are close to one.

To test the statistical significance of the likelihood ratio, one specifies a power and then B.E.St. uses binomial distributions to test whether the $p(D_i|H_1)$ is statistically different from $p(D_i|H_0)$. The following gives a brief overview. Let:

$$d = 2[p(H_1)p(H_0)^{1/2}\{arcsin[p(D_i|H_1)^{1/2}]-arcsin[p(D_i|H_0)^{1/2}]\}$$

Then the effect size, D, can be calculated as:

$$D = (e^{2d}-1)/(e^{2d}+1)$$

Kramer and Thieman provide tables for the number of cases needed to detect the effect size D in a binomial distribution at different levels of power.[17] B.E.St. compares this required number of cases to the observed number of cases in the knowledge base. If the observed number is lower than the required number of cases, then there is insufficient data to detect the difference between $p(D_i|H_1)$ and $p(D_i|H_0)$; therefore, the likelihood ratio is not used in predicting the target event. In these cases, B.E.St. will attempt to create likelihood ratios by combining neighboring responses to the same question in order to create a large enough sample to justify including the question.

---

16.  T. Chard, "Self Learning For A Bayesian Knowledge Base. How Long Does It Take For The Machine To Educate Itself," *Methods of Information in Medicine*, 26: 185-188 (1987).

17  H. C. Kramer and S. Thieman, *How Many Subjects: Statistical Power Analysis and Research*, Sage Publications, Inc., Newbury Park, California (1987).

## 4.2 DYNAMIC SEQUENCING OF QUESTIONS

B.E.St. changes the sequence of its questions based upon the answers it has previously received; this process is typical of many clinical interviews. Each time a question is asked, B.E.St. recalculates the likelihood ratios associated with the answers to the remaining questions. For the next inquiry, it chooses the question with a response having the likelihood ratio with the most extreme value. In essence, B.E.St. asks the questions with rare but very informative answers first.

Researchers have suggested alternative methods for sequencing questions. One group of researchers rely on the expected impact of the questions (the criterion is defined as a function of both the probability and the likelihood ratios associated with the answers to a question).[18] The second group of researchers sequence questions by asking questions with maximum conditional correlation first).[19]

The approaches described above are unlike the approach used by B.E.St.; both the computational requirements and the sequencing of questions differ. We prefer the B.E.St. approach because questions with rare but informative answers area asked at the beginning of the interview, when there is less chance that the knowledge base has been reduced (due to dependencies), and when it is more likely that sufficient data exist to use the answers to the informative questions.

Another advantage of the B.E.St. approach to sequencing is that it is computationally simple. Because the likelihood ratios are already calculated for the inferential task, no new computations, e.g. calculation of correlations among questions, are needed.

## 5. CONFIDENCE BUILDING

B.E.St. attempts to increase the user's confidence in the system's advice by explaining the reasons for its predictions. It lists clues with likelihood ratios larger than one as being for the prediction, and clues with likelihood ratios less than one as being against the prediction. A number of studies have shown that explaining one's reasoning increases the acceptance of one's conclusions by others. In one study, subjects listed

18   L. R. Bigongiari, D. F. Preston, L. Cook, S. J. Dwyer, S. Fritz, D. G. Fryback, J. R. Thornbury, "Uncertainty/Information As Measures of Various Urographic Parameters: An Information Theory Model Of Diagnosis of Renal Masses," *Investigative Radiology*, 16: 77-81 (1981).

19   D. G. Fryback, "Bayes Theorem And Conditional Nonindependence Of Data In Medical Diagnosis," *Computers and Biomedical Research*, 11: 423-434 (1978).

persuasive pro an con arguments for their judgments.[20] Other individuals, after being exposed to the same arguments, changed their attitudes to conform with the judgments of the original subjects. This study revealed that explaining a judgment, in terms of arguments for and against it, induces others to accept the judgment. More direct evidence comes from the work of Erdman.[21] He developed a computer consultation program for advising physicians about depression. Half of the physicians received only computer advice, while the other half received both the advice and explanations from a computer. Erdman found that the explanations helped some groups of users decide whether the computer's advice was reasonable.

B.E.St. also attempts to increase the user's confidence by providing anecdotal evidence in support of its predictions. The system searches its knowledge base for cases similar to the current case and prints the five most similar cases. Similarity is defined by the following formula:[22]

$$\text{Similarity of case A and B} = D_{A,B}/(D_{A,B}+D_{A,Not\ B}+D_{Not\ A,B})$$

Where $D_{A,B}$ is the number of clues in both case A and case B, $D_{A,Not\ B}$ is the number of clues not in case A but in case B, and $D_{Not\ A,B}$ is the number of clues in case A but not in case B. This allows the user to use the printed cases as anecdotal evidence in support of the system's predictions.

## 6. MYOCARDIAL INFARCTION APPLICATION

### 6.1 DESCRIPTION OF TEST DATA

To evaluate the effectiveness of B.E.St., we used it to predict mortality from myocardial infarction. Under a separate grant from the Health Care Financing Administration, we collected data on approximately 1100 patients with myocardial infarction.[23] For each patient, this data base contained clues describing the patients' condition on admission as well as one variable describing the patients' condition upon discharge from the hospital (coded as alive or dead).

---

20  E. Burnstein, A. Vinokur, and Y. Trope, "Interpersonal Comparison Versus Persuasive Argumentation," *Journal of Experimental Social Psychology*, 9: 236-245 (1973).

21  H. P. Erdman, "The Impact Of Explanation Capability For A Computer Consultation System," *Methods of Information in Medicine*, 24: 181-191 (1985).

22  A. Tversky, "Features Of Similarity," *Psychology Review*, 84: 327-352 (1977).

23  F. Alemi, J. Rice, and R. Hankins, "Predicting In-Hospital Survival Of Myocardial Infarction," *Medical Care*, 28(9): 762-775 (1990).

The questions/variables used to predict in-hospital mortality are given in Table 1. Other authors have classified these factors into groups, based on their relative deviation from normal values.[24, 25] Because B.E.St. currently works only with discrete variables, we used the groupings suggested by these authors.

A recent study found that the above set of questions/variables are as predictive of in-hospital mortality in patients with myocardial

| Table 1: Questions Used to Predict In-Hospital Mortality | | |
|---|---|---|
| 1. Temperature | 7. Sodium | 13. Age |
| 2. Mean Arterial Pressure | 8. Potassium | 14. Chronic Health |
| 3. Heart Rate | 9. Creatinine | 15. Surgical Treatment |
| 4. Respiratory Rate | 10. Hematocrit | 16. Emergency Admission |
| 5. Oxygenation | 11. White Blood Count | |
| 6. Arterial pH | 12. Coma Score | |

infarction, as are five other sets of questions/variables.[26] Therefore, our choice to focus on the above clues is as reasonable as if we were to focus upon any other set of clues.

The performance of B.E.St. can be compared to that of optimal statistical procedures such as logistic regression. In logistic regression, a dichotomous variable like mortality is regressed on other variables. Since both logistic regression and B.E.St. use the same set of questions/variables and the same data, the performance differences are only due to the method that is chosen for making inferences.

24  W. A. Knaus, E. A. Draper, D. P. Wagner, J. E. Zimmerman, "APACHE II: A Severity Of Disease Classification System," *Critical Care Medicine*, 13(10): 818 (1985).

25  W. A. Knaus, J. E. Zimmerman, D. P. Wagner, E. A. Draper, D. E. Lawrence, "APACHE -- Acute Physiology And Chronic Health Evaluation: A Physiological Based Classification System," *Critical Care Medicine*, 9: 591-597 (1981).

26  F. Alemi, J. Rice, and R. Hankins, "Predicting In-Hospital Survival Of Myocardial Infarction," *Medical Care*, 28(9): 762-775 (1990).

## 6.2 METHOD OF COMPARISON

B.E.St. uses two thirds of the data for its knowledge base, and the remaining data, referred to as the "hold out" sample, were used in the cross-validated evaluation of the system. Similarly, a logistic regression would require the same two thirds of the data for parameter estimation and the same hold out sample for evaluation. Two different logistic regressions were run. In one regression, all 16 variables were forced into the model. In the other regression, stepwise selection was used, and only variables that were found to be significantly related to mortality were included.

We compared the performance of B.E.St. and the logistic regression using Receiver Operating Curves (ROCs). An ROC is constructed by assuming several cutoff points or scores for predicting whether one will live or die. Patients with a score exceeding a cutoff point are predicted to die, while those with a score below the cutoff point are predicted to live. By experimenting and changing the cutoff point, a curve may be constructed and an optimal point may be found where differences between observed and predicted survival are minimized. The two ends of the curve show two extreme strategies for choosing a cutoff point. If one assumes all patients will live, in effect he/she will accurately predict all cases discharged alive, but will not predict all of the deaths; such an assumption would result in perfect sensitivity but no specificity. On the other hand, if one assumes all patients will die, he/she will accurately predict all cases who die, thus achieving perfect specificity but not sensitivity. A straight line between these two extreme points shows the sensitivity and specificity of a random prediction of those cases who will live. The areas between ROCs and this line show the predictive ability of each approach. The larger the area under the curve, the more accurate the index.

We estimated the area under the ROCs using procedures recommended by Hanley and McNeil.[27] Because the approaches are compared using the same hold out sample, and are related to the same outcome, the areas calculated for each approach were interdependent. To correct for these interdependencies, we used a method also suggested by Hanley and McNeil.[28, 29]

---

27 J. Hanley and B. McNeil, "The Meaning And Use OF The Area Under A Receiver Operating Curve," *Diagnostic Radiology*, 143(1): 29-36 (1982).

28 B. J. McNeil and J. Hanley, "Statistical Approaches To The Analysis Of Receiver Operating Characteristic Curves," *Journal of Medical Decision Making* 4(2): 137-150 (1984).

29 J. A. Hanley and B. J. McNeil, "A Method of Comparing The Areas Under Receiver Operating Characteristic Curves Derived From The Same Data," *Radiology*, 148: 839-843 (1983).

## 6.3 RESULTS OF COMPARISON

Table 2 shows the results of the logistic regression on the learning data base. Note that only 7 out of the 16 variables had a statistically significant relation to mortality. Thus, according to the logistic regression, a model including only the 7 significant variables would be as predictive as a model with all 16 variables.

| Table 2: Variables Predictive of Mortality in Logistic Regression | | |
|---|---|---|
| Variables | All Questions / Variables Model | Stepwise Model |
| Intercept | +4.9186* | +4.9987* |
| Temperature | +0.1504 | |
| Arterial Pressure | +0.3477* | +0.3301* |
| Heart Rate | -0.0054 | |
| Respiratory Rate | -0.2094 | |
| Oxygenation | -0.2270* | -0.2603* |
| Arterial pH | -0.0381 | |
| Sodium | -0.0170 | |
| Potassium | -0.1096 | |
| Creatinine | -0.3759* | -0.3871* |
| Hematocrit | -0.4614* | -0.4343* |
| White Blood Count | -0.7648* | -0.8801* |
| Coma Score | -0.1330 | |
| Age | -0.4772* | -0.4892* |
| Chronic Health | -0.1446 | |
| Surgical Tx | -0.8450* | -0.9351* |
| Emergency Admission | -0.7403 | |

Logistic regression finds the single optimal combination of variables that explain the data. Similarly, B.E.St. also uses all of the variables and finds subsets of questions/variables that are most predictive. However, the subsets of questions/variables used in the B.E.St. model vary from case to case. In some cases, questions/variables are ignored because they are left unanswered or because they fail the

sample test requirements that were previously explained. However, variables ignored in one case may be used in another.

Table 3 shows the use of the 16 questions/variables by B.E.St. in predicting mortality using the learning data set. The right hand columns report the impact of each question in cases where these questions were

| Table 3: Use of Questions/Answers by B.E.St. in Predicting Mortality | | | |
|---|---|---|---|
| Variables | Percent Used | Mean Impact When Used | Standard Deviation |
| Temperature | 18.75 | 2.04 | 0.93 |
| Arterial Pressure* | 44.13 | 1.36 | 0.08 |
| Heart Rate | 5.75 | 1.40 | 0.39 |
| Respiratory Rate | 96.25 | 1.54 | 1.19 |
| Oxygenation* | 7.13 | 2.61 | 2.05 |
| Arterial pH | 20.00 | 2.07 | 1.69 |
| Sodium | 57.50 | 1.06 | 0.08 |
| Potassium | 2.25 | 3.33 | 2.55 |
| Creatinine* | 47.50 | 2.03 | 1.62 |
| Hematocrit* | 94.38 | 1.20 | 0.32 |
| White Blood Count* | 93.75 | 1.49 | 0.82 |
| Coma Score | 95.00 | 1.42 | 1.57 |
| Age* | 95.25 | 1.59 | 0.47 |
| Chronic Health | 0.00 | - | - |
| Surgical Tx* | 97.88 | 1.98 | 0.35 |
| Emergency Admission | 0.00 | - | - |

\* — Variables which were statistically significant in logistic regression reported in Table 2.

used (for a definition of how impact was calculated see the Terminology section). Also note that the average impact of each question changes from case to case. This is due to the fact that different answers to the questions have different likelihood ratios, and that the likelihood ratio associated with an answer changes as the knowledge base is reduced. Because of the variation in the impact of each question, the standard deviations for the impact of each question are also reported. The questions/variables with the largest average impacts are the most

informative when they are used. A comparison of Tables 2 and 3 highlights the differences in how the questions/variables are used in the two different approaches.

For example, note that Potassium is almost never used (2.25%), but when it is used, it has a large impact on the prediction of mortality (Mortality odds are changed by a ratio of 3.33). In contrast, Potassium does not appear to be significantly related to mortality in the logistic regression model.

The ROC curves associated with B.E.St. and with the stepwise logistic regression using the hold-out sample are shown in Figures 2 and 3. The area under the B.E.St. ROC curve was 81.19% and the area under the Stepwise Logistic Regression ROC curve was 76.54%. The difference between these two areas is statistically significant at an alpha level of 0.07 ($z = 1.80$), suggesting that B.E.St. constructed a more accurate model than the Stepwise Logistic Regression.

## 7. DISCUSSION

One of the major strengths of B.E.St. is the manner by which it accounts for dependencies. Most statistical approaches attempt to reduce the number of variables in a model until an optimum number remains; variables that are interdependent are dropped out of the model. B.E.St. takes an opposite approach. All variables are used by the system, including dependent variables. Therefore, B.E.St. is more robust to missing information, a problem for the optimal statistical approaches. B.E.St. will only ask one of two redundant questions, since asking the other question would not be informative. This procedure has no effect on the data collection time. Thus, compared to the optimal statistical approaches, B.E.St. is more robust and does not have any additional data collection requirements.

Redundant systems are designed to function very much like human beings and experts. For example, words in a singe sentence are often redundant. Such redundancy in language ensures that a sentence can still communicate the intended meaning even if some words are omitted. By allowing for redundancy, B.E.St., in effect, simulates the robustness of an actual expert.

In this study, one of the surprising findings was the relative performance of B.E.St. and the logistic regression. Both approaches derive their parameters from the data, but B.E.St. was more accurate than the logistic regression. Unlike B.E.St., logistic regression ignores variables that are very informative but rare. For example, the presence of coma is very informative in predicting mortality, but the overwhelming majority of patients are not in a coma. Therefore, the coefficient for this variable is not statistically significant in the logistic regression. In contrast, B.E.St. uses this variable considerably.

We have not compared B.E.St. to other statistical approaches like Automatic Interaction Detection. In addition, our findings are based on one data base and may not be generalizable.

# B.E.ST
## ROC AREA = 81.19

Figure 2
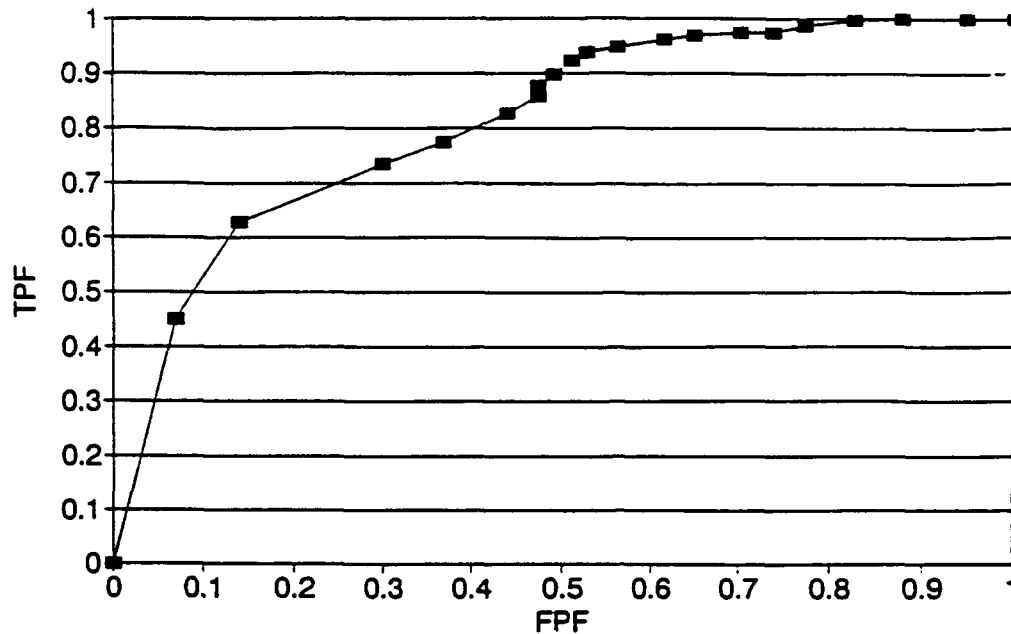


# LOGISTIC REGRESSION
## ROC AREA = 76.54

Figure 3
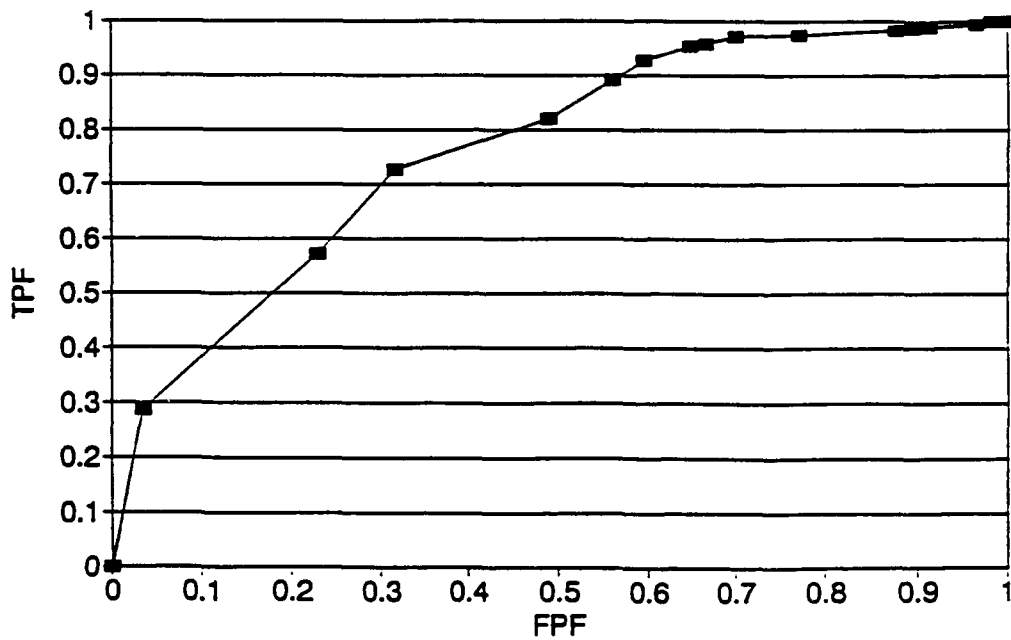
# A NATURAL LANGUAGE GENERATION SYSTEM
## FOR A
# HETEROGENEOUS DISTRIBUTED DATABASE SYSTEM

Bipin C Desai

Georgios Ioanni Kouklakis

Computer Science, Concordia University

Montreal, H4B 1R6, CANADA

## ABSTRACT

In this paper we describe the design issues of a database independent natural language generation system (NLG system) targeted for non-technical users. The NLG system generates output in the form of a coherent, consistent and stylish text. A prototype NLG system which addresses these issues has been built for integration into a multilevel interface for a Heterogeneous Distributed Database Management System.

## INTRODUCTION

Database systems are used by different kinds of users for various purposes. The users can generally be classified into two different types: technical and non-technical users. The latter pose simple requests and have very little knowledge of the database system, other than the conceptual organization of the objects they want to query.

To bridge the gap between the casual users and a complex database system, a natural language interface (NLI) is required. The principal role of a NLI is that of an intermediary which encodes and translates information in natural language (NL). This provides a means of communication between the machine and the casual user.

The use of NL, as the vehicle for communication, is very important for the following reasons[20]. NL is the common mode of communication in everyday life; using it requires no special training. NL frees the user from knowing how a formal language is used and how information is stored and processed. Natural language is however, syntactically and semantically ambiguous. These ambiguities cannot be resolved completely by a NLI system[7].

Many natural language processing(NLP) systems have been developed in recent years. Some examples of these are ASK[28], Data Talker[20], Eufid[27], TQA[6], KID[16], Datalog[14], TEAM[13], KAMP[2], [3], MUMBLE[24], PHRED[17], PAULINE[15], TEXT[22], Ana[18], Yh[12] and Adorni[1]. These sytems are used for a wide range of applications, such as, machine translation, story comprehension, language learning assistance, database front-ends, and tutoring systems.

This work will focus on the design and implementation of a NLG system. A model for the system is proposed in this work. The production of text generation is viewed as a mapping from meaning to text, through a series of transformations. A prototype system has been developed using this approach for integration with a Multilevel Interface to an Heterogeneous Distributed Database

Management System (MIDBMS) environment[10]. The MIDBMS environment provides its users with a multilevel interface using different languages such as natural or formal query languages. The interface consists of four levels, based on the users knowledge of the system: the NLI, SQL, General Mapping and Query Language(GQML)[25], and local database query language interfaces levels.

The conceptual architecture of the system is shown in Figure 1. At the NLI level, the users pose their queries using natural language. This level is targeted to the users of the system who merely want to make some casual queries. However, the users are assumed to have a knowledge of the underlying database schema, since the system does not support a dialogue with the user to clarify his/her questions regarding the database structure or the way to form queries.

At the second level, SQL is provided as the language to access the database. The database users who are experienced with the syntax of the language and the structure of the database can form their requests directly in SQL, skipping the NLI level. This saves computational time and results in a more efficient processing. The request in SQL can be stated in a clear and unambiguous way, compared to a natural-language request which may result in a misunderstanding due to the vagueness of the natural language. This level can be considered as the bridge linking the natural language and GQML. The GQML operates on a Heterogeneous Distributed Database Management System(HDDBMS). In this way, a higher level interface is provided over several different types of databases, each of which has its own native query language.

Since GQML is operating on a HDDBMS environment, it has special operations related to the mapping among the heterogeneous database schema which are not provided in the SQL. The processing of the GQML includes interpreting the global query into a set of local queries. These are sent to the corresponding local DBMS, collecting the database results from each of the local database systems which are involved in the global query processing, translating and combining all the collected database results and relaying to the previous level. Consequently, it appears to the database users that they are interfacing with a single conventional database system rather than a set of different DBMS products. Each component DBMS of the HDDBMS provides a local query language and a local user could use it to query the individual database system.

At the highest level of the interface, there is the lack of a natural language response from the system. The motivation for the proposed NLG system is the realization that a NLI should produce a response which must consist of natural language sentences, rather than a formal output. For non-technical users, interpreting a complete text is much easier and more natural than understanding a structured answer from the computer. Thus, the requirement of the NLG system is to produce stylish text, which is coherent, consistent and easily comprehensible. The NLG system can also be used to provide feedback to the user. This can be done by paraphrasing the user's request and sending it back to the user. The paraphrase can be generated from the SQL query by applying the transformational steps described in the Section 3 and turning a declarative sentence into an imperative.

This paper is organized as follows. In section 2, a brief discussion of the components of the NLI systems is presented. Different approaches for syntactic and semantic processing, and query interpretation, are addressed. In section 3, the model of the proposed NLG system is described. Section 4 we give a detailed
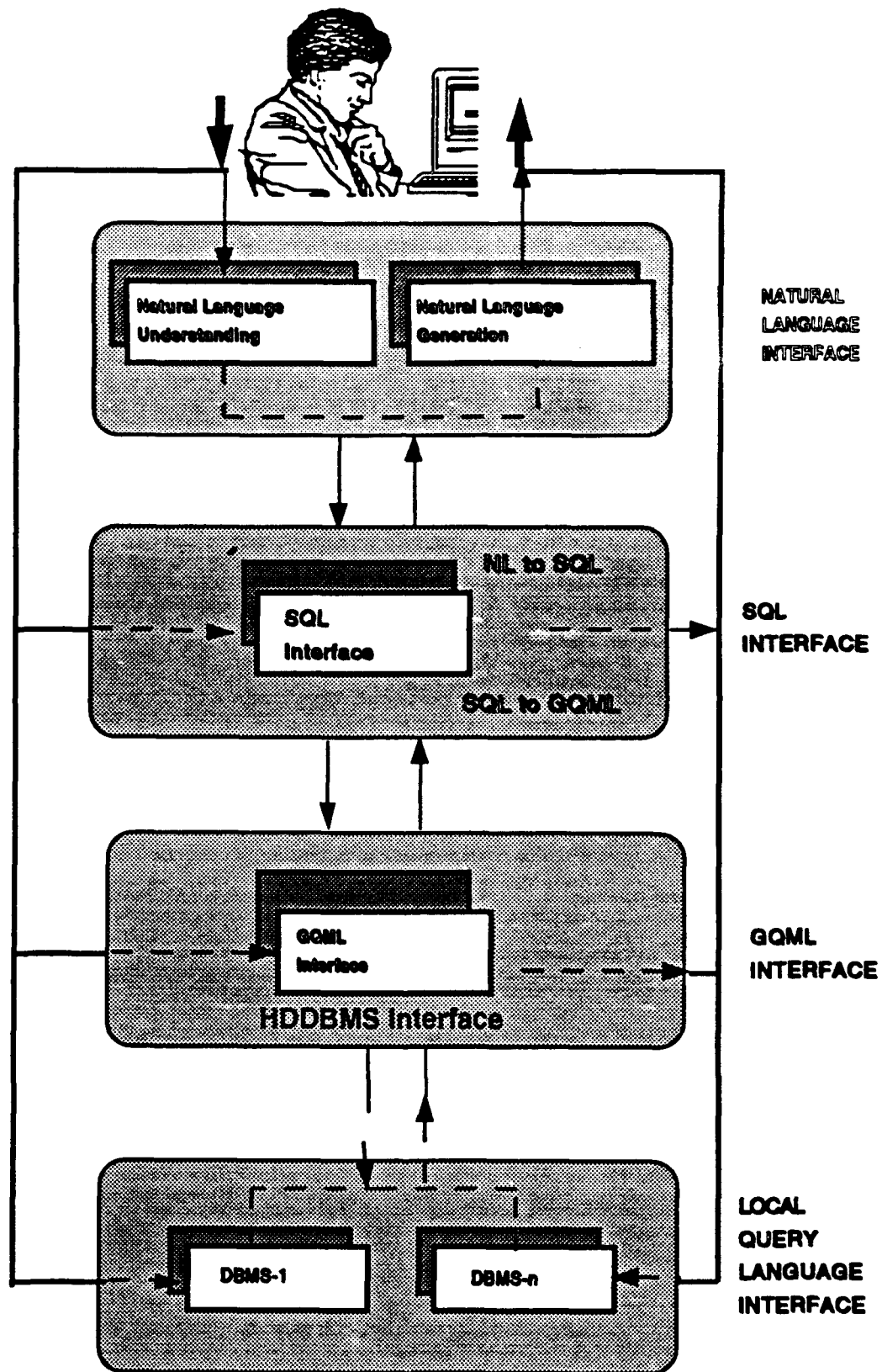
Figure 1 Integration of NLG system into MIDBMS
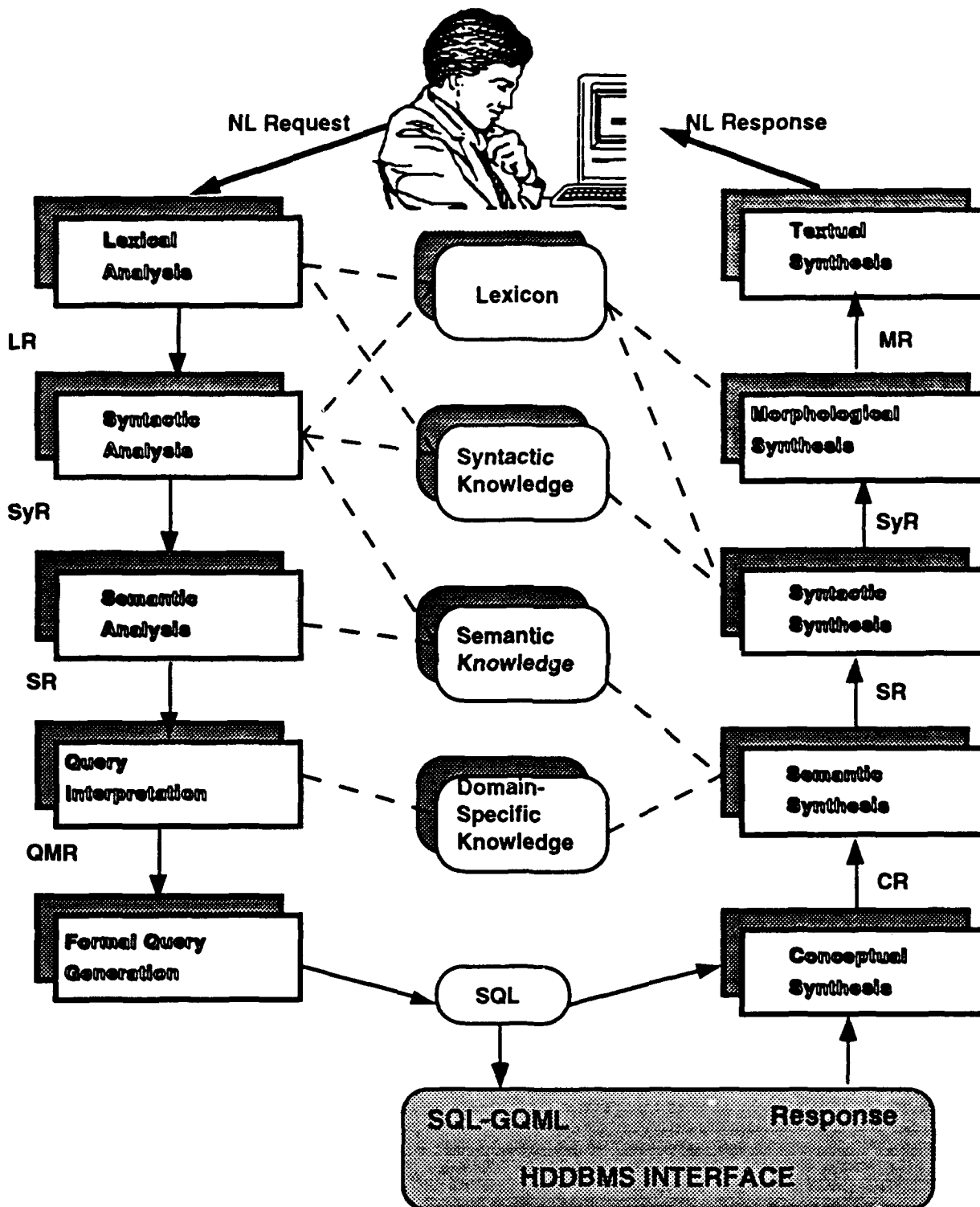
Figure 2 NLI Architecture of MIDBMS

example. Finally, section 45 gives the conclusion and further development guidelines.

## 2. Components of a NLI

The integration of NLG into the MIDBMS is shown in Figure 1 The NLI consists of ten processing phases, of which five are "down" phases and five are "up" phases (Figure 2). In the following discussion, we elaborate on the functions of the NLI modules.

NLP systems use language information in many different forms to communicate effectively with the end-user. In the context of computational linguistics, natural language is described in terms of its morphology, syntax, semantics and pragmatics[19]. In order to process a user's request and respond appropriately to it, the NLI to a DBMS assigns an internal interpretation to the natural language input and generates from this representation a response that is natural as well. To do so, it integrates information from a number of sources of knowledge. A NLI system must embody not only a grammar, a lexicon, and morphological rules, but also a parser and translator, to effectively analyze input sentences and generate responses in accordance with grammatical and morphological rules.

The proposed system is designed to operate under the MIDBMS environment, and the lexicon and extended Entity/Relationship Model proposed in [10] are used. An entry in the lexicon, called lexeme, is defined with its syntactic and semantic properties. The syntactic part of the definition contains all the necessary syntactic categories of the lexeme, such as noun, verb, preposition etc. The semantic part contains the semantic category of each lexeme, for example, animate, abstract object, and so on.

The parser verifies that the sentence is syntactically well-formed and determines its linguistic structure. During the parsing process, the linguistic relations such as subject-verb, verb-object, and noun-modifier are determined. These relations provide the framework for semantic interpretation during the natural language understanding(NLU) phase and response generation in the NLG phase.

In the NLU phase, two basic strategies for parsing exist: top-down and bottom-up[26]. Top-down parsers construct the parse-tree by starting at the top and working downward. They start with the symbol for "sentence" and recursively expand it, until the parse-tree contains pre-terminal symbols which can be checked against the lexicon.

Top-down parsers often impose constraints on the allowable grammar rules. For instance, a left-to-right top-down parser requires that the grammar be put in a form which avoids the use of recursive left-branching rules, which could result in infinite processes during parsing.

Bottom-up parsers start with the input words and develop the parse-tree from the bottom-up, by replacing right-hand-side patterns with those from the left-hand-side. They end when all that remains is the "sentence" symbol.

As opposed to top-down parsers, bottom-up parsers avoid the backtracking problem, and the generation of infinite parse-trees. However, they still produce undesirable results such as the generation of all possible sentences.

The input of the Lexical Analysis module is a natural language sentence. The lexical analyzer searches the lexicon to find the lexical entries for the words appearing in the sentence. In natural language, a word may have several syntactic or semantic definitions. As a result, a sentence could have several lexical interpretations. The task of the Lexical Analyzer is to provide associativity between each word and its syntactic and semantic definitions.

For each word, all its possible lexical interpretations are constructed in the form of syntactic markers. Each lexical representation(LR) is submitted to the Syntactic Analysis module for further processing. The syntactic analyzer applies grammar rules to the LR to verify that the sentence is grammatically legal. If the sentence fails to be parsed, the next lexical interpretation is processed, and so on, until a suitable interpretation is successfully parsed into a Syntactic Representation(SyR), or there are no more lexical interpretations.

If the sentence is parsed, a parse-tree is produced. The parse-tree contains lexical markers, such as noun and verb and phrase markers, such as, noun phrase and verb phrase.

Since it is possible for a syntactically legal sentence to make no sense because of its semantic disagreement, Semantic Analysis is performed. Semantic processing is needed to translate the parse-tree into a semantically-legal interpretation in the NLU phase and assign semantic roles to formal (database) objects in the NLG phase. The output of the Syntactic Analysis module is accepted by the Semantic Analyzer module and semantic restriction rules are applied.

If the semantic analysis fails, the next SyR is processed until a semantically legal interpretation is encountered or the SyR's are exhausted. A semantically legal interpretation is recorded as a Semantic Representation(SR) and it represents the deep structure of the sentence.

The Query Interpretation module accepts the SR's and performs the interpretation of the query. If the query interpretation fails, the next SR is examined until one is interpreted correctly or the SR's are exhausted. If the query interpretation phase succeeds, the SR which is linguistic-oriented, is transformed into a query meaning representation(QMR) which is query specification oriented. QMR fills the gap between the general meaning representation and the formal query language.

The task of the Formal Query Generator is to map the QMR into an SQL query. The SQL query is translated into a GQML query which is then handled by the HDDBMS system.

For a generally applicable natural language interface the main concern is that of embodying the knowledge in order to establish the associations between the semantic notions and the database notions. For the purpose of portability it is reasonable to separate knowledge about English words from database attributes. For this reason the knowledge can be separated into two parts: one which is domain-independent and another which is domain-dependent.

To model the database, the Entity-Relationship(E-R) Model is used. According to this model, the entities represent distinguishable objects, and the relationships represent the associations between these entities. The entities and relationships have a number of attributes, and each attribute has an associated domain. For example, a <University> database which consists of three entities, namely <Student>, <Course> and <Teacher>, and two relationships <Stu-Cour> and <Cour-Teach> is represented using the E-R model shown in Figure 3. In the E-R model, the arcs between an entity or a relationship and their attributes, and between an entity and a relationship represent the existence of various associations. They do not, however, indicate what kind of association it is, because they lack the semantic content. To overcome this drawback and enhance the model, an explicit semantic meaning is assigned to each arc in the form of a transitive verb. Figure 3 shows the extended E-R model for an University database.
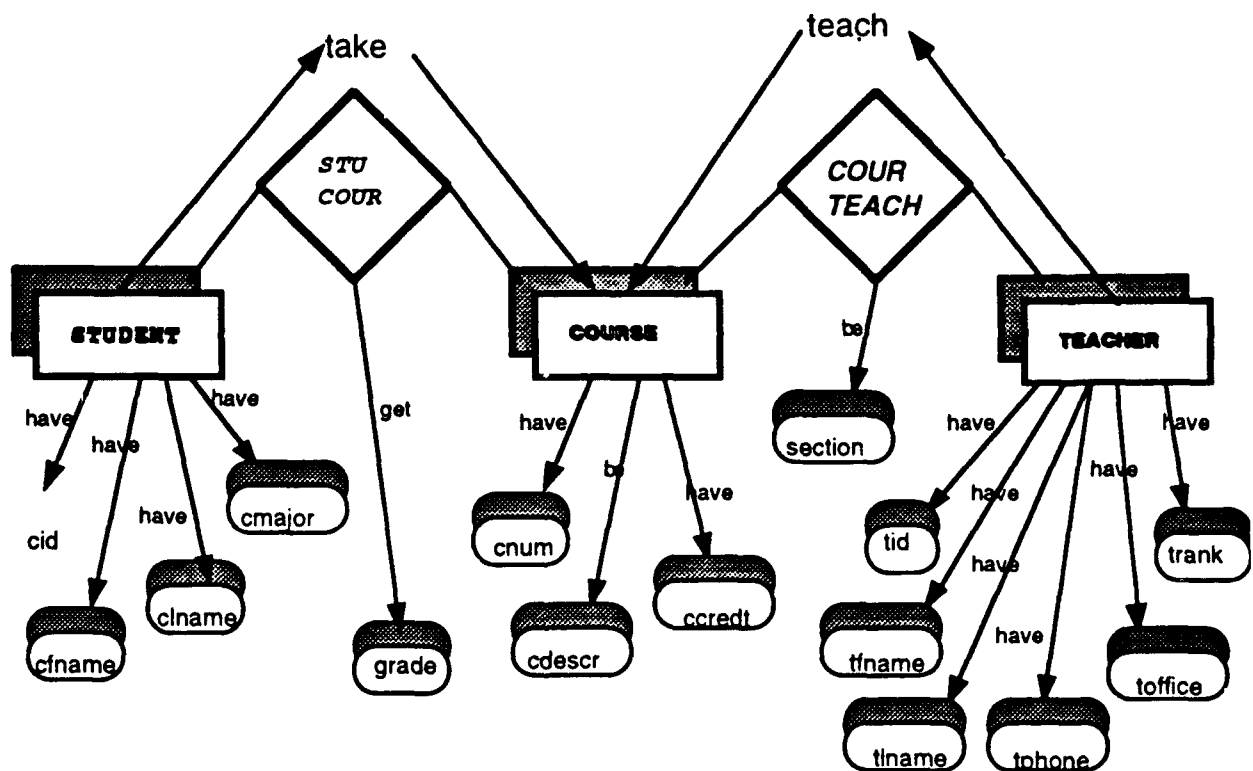


Figure 3   Extended E-R Model for an University Database

Finally, a preposition is assigned to each arc, which is illustrated by an arc from attribute to an entity or a relationship. For example, the <Teacher> entity may be modifiedas shown in Figure 4.
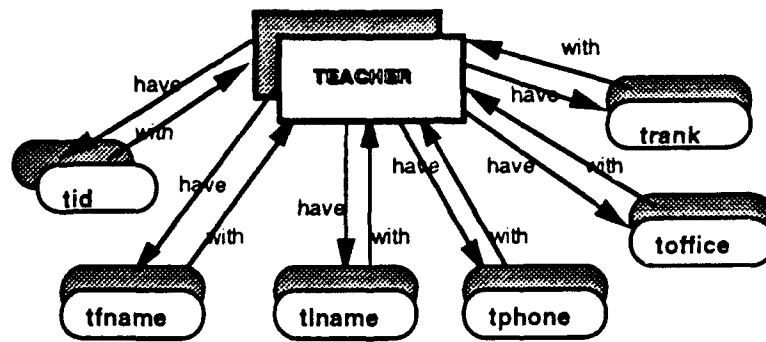
Figure 4   Teacher Entity

Many researchers have attempted different approaches to represent the deep meaning of a sentence. Commonly used representations include conceptual graphs, conceptual dependencies, case-frames and logic-representations. The most influencial work among these is the Case Grammar developed by Fillmore[11]. Case Grammar is semantically-oriented and establishes a representation for expressing the deep meaning of the sentence to be generated or analyzed. The verb plays an important part in building the semantic representation, since it defines relationships between the subject, object and other syntactic components of the sentence. For each verb, a case frame specification is set up to indicate which case is required, optional or not allowed.

The case frame representation is used for the description of the transitive verbs. Figure 5(a) shows the domain-independent definition of the verb *teach*, and Figure 5(b) shows the domain-dependent definition of *teach* within the context of the university database. This separation of knowledge into domain-dependent and domain-independent parts is very important for making the system transportable and easily expandable.

teach(agent:   optional
    action:  teach
    instrumental:   not_required
    dative:  optional
    neutral:   required
    locative:  optional)

teach(agent:   teacher
    action:  teach
    instrumental:   null
    dative:  null
    neutral: course
    locative: campus)

Figure   5(a)   Domain-Independent
              Definition of *teach*

Figure   5(b)    Domain-Dependent
              Definition of *teach*

At the query interpretation stage of processing, a mapping between the semantic primitives and the database attributes is established. In the NLU phase the task of the query interpreter is to map the meaning representation structure obtained from the semantic analysis onto a query meaning representation scheme. This creates a query-oriented representation which can be processed by the query language translator. During the NLG phase, the query along with the answer obtained from the database, is mapped to the semantic structure. With this mapping, semantic markers are assigned to the database attributes.

## 3. NLG of Database Response

The requirement of the NLG system proposed in this work, is to generate a natural, coherent and stylistically satisfying English text, in the form of one or more sentences. The generated text is a response to a user's query, formed using natural language. The NLG system is database-independent, i.e., the generation process is not affected by the change of the domain of the database. To model such a system, the Meaning Text Model formalism[23] is adapted in combination with Chomsky's phrase-structure approach[4]. The knowledge sources needed for the generation process, are the lexicon and the semantics expressed in the extended E-R model of the database.

The first phase of the NLG process is the Conceptual Synthesis module. It generates CR: the conceptual representation, of the response to be generated by the NLG system. This module is the interface between the NLG system and the GMQL.

The input to the conceptual synthesis process is a SQL query $Q$ (SC, FC, WC) and a database response $\Re$. Here SC is the select clause, FC is the from clause and WC, the where clause of the query. The response $\Re$ is in the form of a table with a column for each attribute in the select clause. The SQL query is taken from the formal query generation module and the the corresponding database response, in table form, from the GQML translator.

CR is a compact form of the query and the response, which preserves the association between an entity or a relationship and its attributes and between an attribute and its values. The CR is the input for the semantic synthesizer module. It is a list created from four sets: S, F, W, and $\Re$, where:

$S = \{a_{ij}| a_{ij} \epsilon SC\}$ i.e., the set of attributes in the select clause of the
SQL query,
$F = \{f_i| f_i \epsilon FC\}$ i.e., the set of entities and relationships in the
from clause,
$W = \{<a_{ij}, v_{ij}> | a_{ij} \theta v_{ij} \epsilon WC\}$ i.e., the set of attribute-value pairs in the where clause (note that the join conditions are not needed and thus they are suppressed),
$\Re$ is the database response.

The elements of the CR list are triples of the form *($f_i$, Sublist1$_i$, Sublist2$_i$)*, where:

$f_i \epsilon F$.
*Sublist1$_i$* = $\{<a_{ij}, v_{ij}> | a_{ij} \epsilon fi \& a_{ij} \theta v_{ij} \epsilon WC\}$
*Sublist2$_i$* = $\{<a_{ij}, \Re[a_{ij}]> | a_{ij} \epsilon f_i \}$

The first sublist, Sublist1, consists of the elements from W such that: a member $<a_{ij}, v_{ij}>$, of the set W, becomes an element in the sublist, if the attribute $a_{ij}$ is an attribute of the $f_i$ relation in the extended E-R model. The second sublist, Sublist2, consists of the elements from the set S and the elements from $\Re$, such that each attribute $a_{ij}$, of the set S which is an attribute of the $f_i$ relation in the

extended E-R model, becomes a candidate for the sublist. For each such attribute, a sublist is created with $\Re[a_{ij}]$.



SR
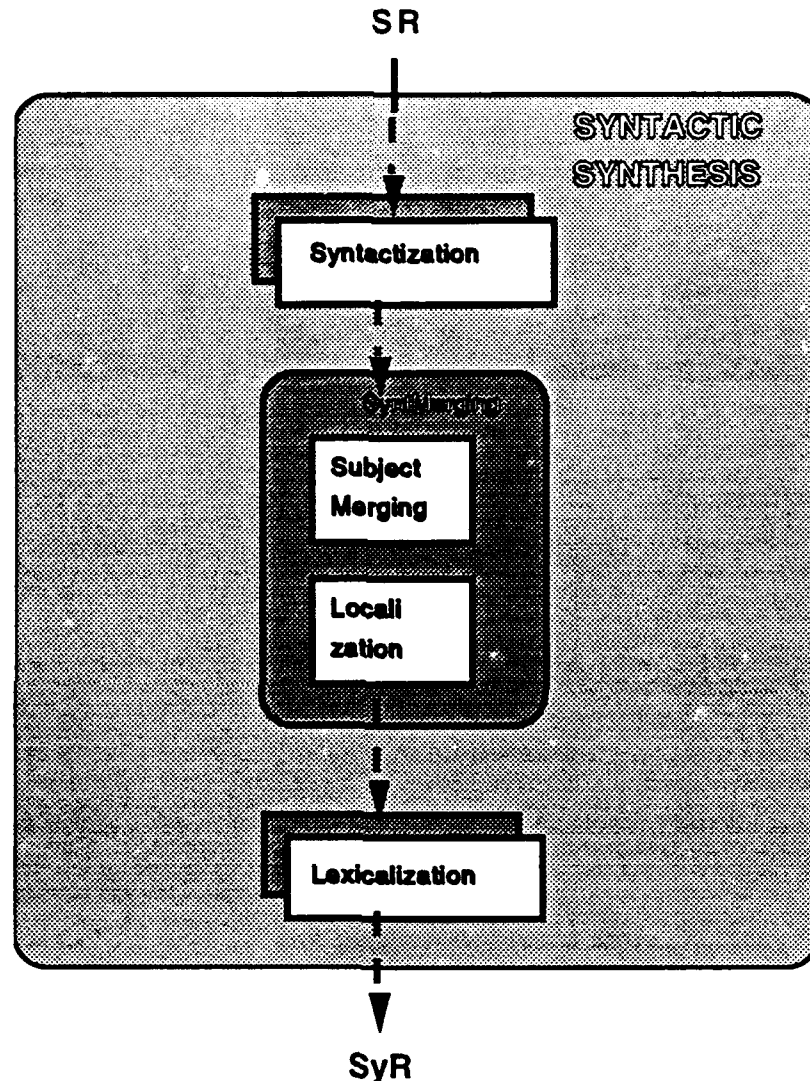
Figure 6 Semantic Synthesis

The input of the second phase, the Semantic Synthesis module(Figure 6), is CR's and its output is a number of semantic representations(SR's), which represent the deep meaning of the sentences to be generated. The process of the SR generation is divided into two subphases: *Semanteme Representation* and *Semantization*. Semanteme representation breaks the CR into smaller units, SRi, which are called *semantemes*. Semantization maps each SRi against the semantic model, in order to assign semantic meaning to the SRi objects.

In the semantization subphase, the principle of decomposability is applied, wherein, the meaning of the whole is constructed out of the meanings of its parts. The meaning of the whole is represented by the CR's and the meaning of the parts by the semantemes. For each SR, the mapping is as follows: (i) Every element of the Sublist1 is mapped onto the corresponding database attribute in

the E-R model. Since each element of the Sublist1 will form a prepositional phrase in the final text, only the 'preposition' arrows are retained. Thus, the mapping of the elements of the Sublist1 to the database attributes retains the 'preposition' arrows and discards the 'verb' arrows. (ii) Each element of the Sublist2 will form the object part of the sentences to be generated. Thus, by mapping each element onto the Extended E-R model, only the 'verb' arrows of the corresponding database attributes are retained and the 'preposition' arrows are discarded. (iii) The attributes of the Extended E-R model which do not occur in either sublist are discarded.

**SR**

Syntactization

Subject Merging

Locali zation

Lexicalization

**SyR**
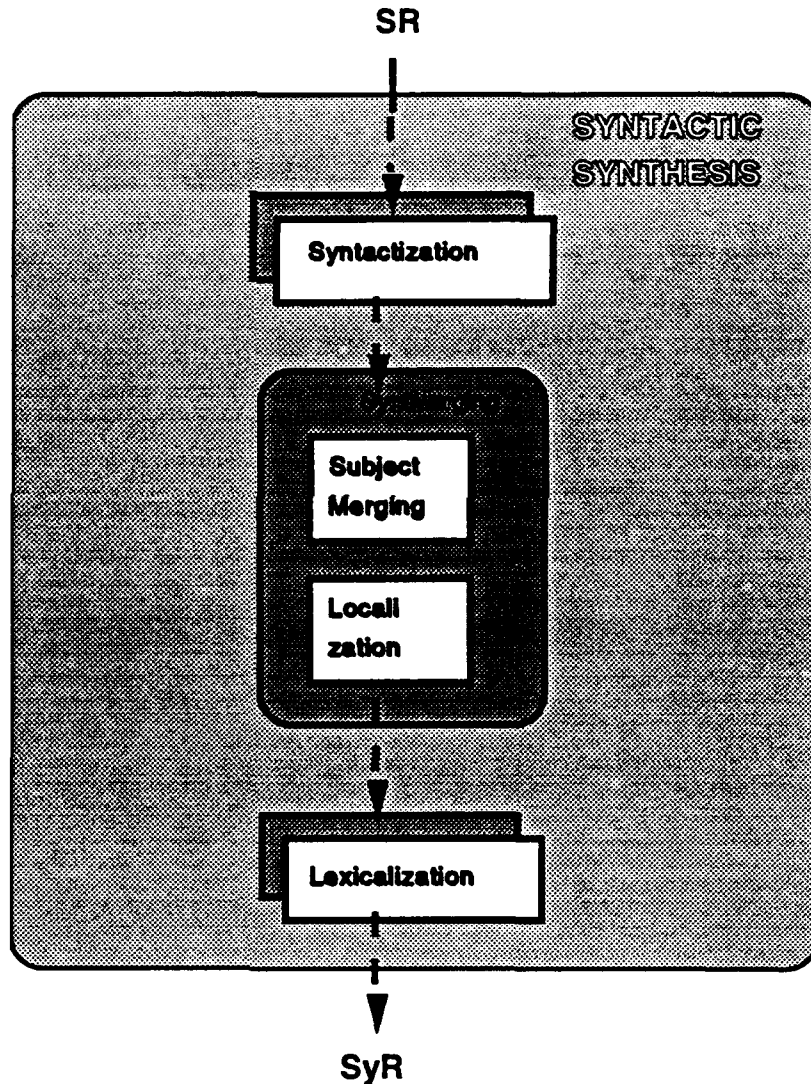
Figure 7   Syntactic Synthesis

In the next phase of the NLG process the intent is to produce the syntactic representation (SyR). It specifies the organization of the sentences based on their meaning. The meaning is represented by the SR's which are created in the previous phase and passed on as the input to this phase. The generation of the syntactic representations is divided into three subphases: *Syntactization,*

*SyntMerging* and *Lexicalization* (Figure 7). Syntactization creates a SyRi for each SRi. SyntMerging merges the SyR's that have common properties. Finally, Lexicalization maps each attribute from the SyR onto a lexicon entry.

The syntactization phase creates three-part structures of the form: <Subject - Prepositional Phrase set (PPset) - Object set>. The PPset is a set of prepositional phrases which we call *global prepositional phrases* (globalPPs), and the object set is a set of zero, one or more objects. The Sublist1 of the SR is mapped onto the globalPP(s) of the SyR, whereas the Sublist2 from the SR is mapped onto the object set of the SyR. Each object may have its own PPset associated with it; we call these the object's *local prepositional phrases* (localPPs).

The second sub-phase operates on the SyR's created by the Syntactization phase. The SyntMerging process has two subphases, namely *SubjectMerging*, which merges the SyR's that have a common subject, and *Localization*, which appends a globalPP of a SyR to a localPP of another SyR. The SubjectMerging process combines two SyR's that have a common subject, into one SyR. The PP and object parts of the new SyR are the union of the corresponding parts of the input SyR's.

During the merging of the object parts of the two SyR's, (one of which corresponds to an entity and the other to a relationship) a process called *object-linearization* occurs. In object linearization, the object set is sorted, such that the first object entry of the new SyR is the first entry of the relationship's object set, followed by the object set from the entity's set, and then the remaining entries from the object set of the relationship.

The localization process appends a globalPP from a SyR to a localPP of another SyR. If there is a SyRi with empty object-set and non empty global PP set, and the subject of the SyRi occurs in the object set of another syntactical representation, SyRj, then localization of the global PP takes place.

Lexicalization, the final subphase of syntactic synthesis, is a mapping between semantic objects and lexemes from the lexicon. Two types of mapping are defined, namely *lexico-semantic* and *phraseologico-semantic* mapping. The lexico-semantic mapping is a substitution of a semantic object by a lexeme from the lexicon. Phraseologico-semantic mapping is very much like a lexico-semantic mapping, the difference being the substituent contains a *phraseme*. A phraseme is a sequence of lexemes, i.e., adjective-nouns, instead of a single lexeme. For example, the mapping of <tfname> into <first name>.

Recall the principle of decomposability: the meaning of the whole is constructed out of the meaning of its parts. Thus, the meaning of the sentence is built up out of the meanings of its constituents. Ultimately, the meaning of the sentence is built up out of the meanings of the lexical items occuring in it. The lexicalization process transfers each database attribute into its corresponding lexical entry. The morphologic synthesis module (Figure 8) generates morphologic representations(MR), an ordered sequence of lexemes in their required forms, out of the SyR's. The process consists of three phases, namely *Sentencialization*, *Lexeme-Transformation* and *Sentence-Generation*. Sentencialization breaks each SyR into a number of MR's, each of them corresponding to a potential sentence. Lexeme transformation implements the transformation of the lexemes to their required forms. Finally, sentence generation

transforms each MR into a sentence-like form by inserting the articles, conjunctions, and punctuation.



**SR**

**MORPHOLOGIC SYNTHESIS**

Sentencialization

Lexeme_ Transformation

Sentence_ Generation

**MR**
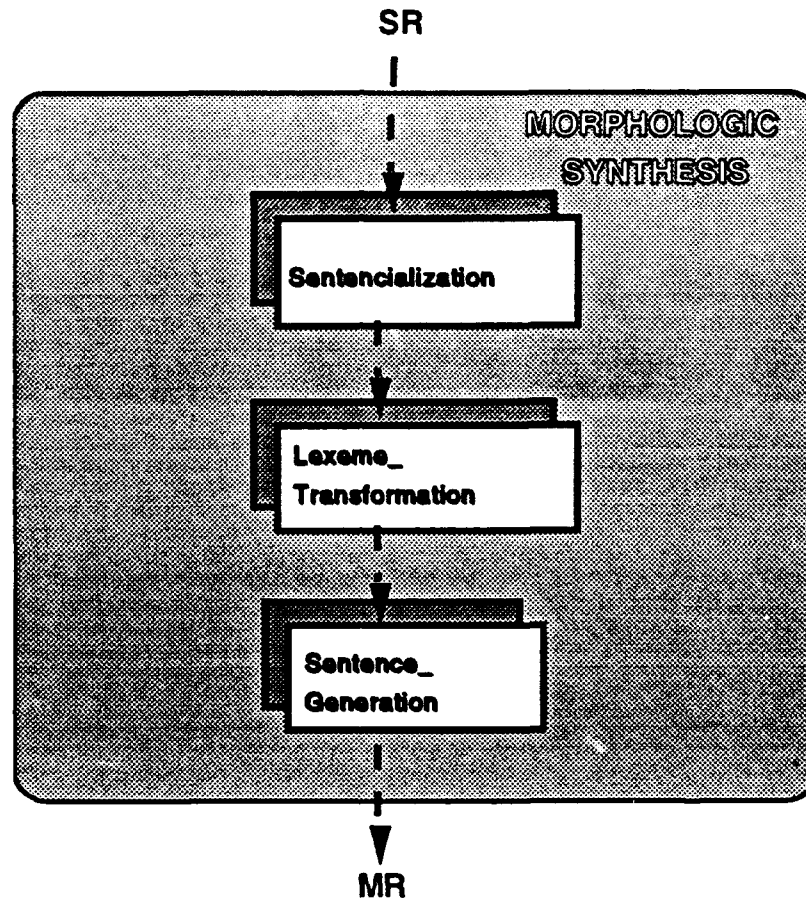
Figure 8 Morphologic Synthesis

The final phase of NLG is Textual Synthesis(Figure 9) which creates the text to be displayed to the users. The intent of textual synthesis is to generate coherent and stylish English text, in the form of one or more sentences. In most cases, MR's created in the previous step are a sequence of independent sentences, rather than coherent text of the final output presented to the user. Textual synthesis reorganizes a sequence of independent sentences into a more readable and stylish text. The TR's are generated from the MR's by applying the textual transformation rules(TTR). These rules combine the MR's with common attributes into compound, anaphoric and elliptical sentences. This results in a coherent, consistent and stylish text.

**MR**
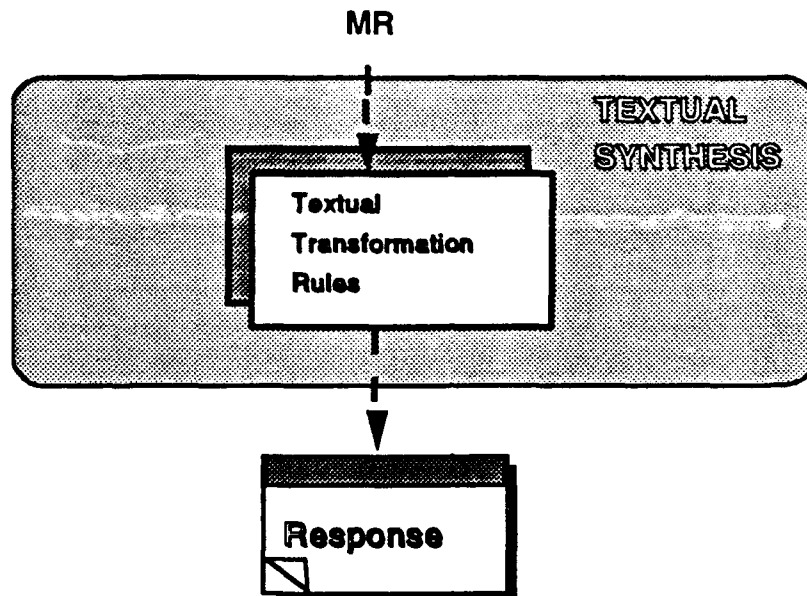


Figure 9 Textual Synthesis

The first two MR's, with the same subject and verb that agree on the number(TTR$_1$), are combined into one by combining the object-value parts. If there is a third MR, with the same subject and verb as the previous ones, it is transformed into its possessive case(TTR$_2$). If there is a fourth MR, with the same subject and verb as before, a pronoun replaces the subject, if the latter is in its plural form(TTR$_3$). In the case where the subject is singular, this step is skipped and the next rule is applied. The problem with the singular case is that, the gender of the noun cannot be determined, in order to use the appropriate pronoun. If there is a fifth MR, with the same subject and verb as before, or if the previous step was skipped, then TTR$_4$ is applied. In this step, each MR is transformed into its possessive case, by applying TTR$_2$. The subject of the sentence is relinquished and an elliptical sentence is generated, retaining only the object from the original MR and the value list which become the subject and object(s) respectively, of the new sentence.

## 54  CONCLUSION

This project proposed a model for a NLG system. A prototype based on this model has been developed in ANSI-C. The choice of the language makes the system easily transportable to different environments which support the C language. Since the HDDMBS (which is at the lowest level of MIDBMS) resides on PCs, the NLG system has been implemented on these machines.

The NLG system is database independent and has been tested using two different database domains. The results have shown that the NLG system meets its design goal: to generate Natural Language as the response of the system to its non-technical users.

The output contains all the necessary information and is in a textual form. The Textual Synthesizer module maintains a textual transformation rule base(TTRB). Currently, the TTRB has four rules to transform the generated sentences into the final text. The generated text consists of one or more sentences in nominative and possessive cases. It may also contain anaphoric and elliptical sentences, based on the common attributes and the number of generated sentences. The text can be further improved by adding discourse rules in the TTRB. Aside from the rules discussed in Section 3, it is possible to add new rules. For example, consider the following three sentences:

| S1: | Subject1 | Verb1 | Object1 |
|-----|----------|-------|---------|
| S2: | Subject1 | Verb2 | Object2 |
| S3: | Subject2 | Verb3 | Object3 |

such that Object1 = Subject2. If the sentence S3 is placed after the S2 it "cuts" the flow of the text. It is rather preferable to place S3 right after S1, as an elliptical sentence by applying the TTR4. Thus, a new rule can be inserted in the TTRB, which implements the above transformation.

At this point the NLG system responds to the users by treating their queries as independent requests. For example, if the user asks for Professor Smith's phone number, the response of the system is "Professor Smith has phone number 327 3943". If the next request is for his office number, the system, will similarly answer, "Professor Smith has office number AD 639". The user might however, expect a more natural and friendly response such as "His office number is AD 639" or simply "It is AD 639". Such system behavior can be achieved if the NLG system can keep track of the most recent history of requests and responses and adjust its behavior accordingly.

A problem arises when the response contains many tuples. For example, the query "List the student names" may generate 100 or 1000 names. Then, it is not feasible to provide a "row" sentence having one subject and a hundred or a thousand objects. Rather, it is preferable to generate the NL text and provide a reference to the corresponding student table, i.e., "The students' names are shown in table 1".

The NLG system is aimed at the non-technical users of the MIDBMS. The users can range from knowledgeable to inexperienced, according to their familiarity with the system. The NLG system should adapt the system's responses to the users' level of expertise in order to behave in an intelligent way. This can be achieved by maintaining a model of the user. In practice, the users are expected to gain some familiarity after using the system for some time. Thus, they may become local experts in specific domains. Therefore, the system should also be able to dynamically change its model of the users by maintaining the relevant parameters that describe their profiles.

One of the goals of the NLG system is to make the user interface more natural. The term natural here means that the system has to behave like humans as much as possible. However, naturalness is not a single characteristic. Rather, it is a collection of attributes that jointly enhance the naturalness of the interface. Providing NLI, in terms of writing text, is one attribute. Using sound is another. Supporting a voice interface for the NLG system will increase the overall re-

trieval efficiency: using voice is faster than writing text, and provides the users with a more familiar environment.

# References

[1] G. Adorni, L. Massone, *Toward a Language - Independent Generator of Sentences* , Applied Artificial Intelligence, 1987.

[2] D. E. Appelt, *Planning Natural-Language Referring Expressions*, Proceedings of the 20th Annual Conference of the Association of Computational Linguistics, Toronto, 1982.

[3] D. E. Appelt, *Planning English Referring Expressions*, Cambridge University Press, Cambridge U.K., 1985.

[4] Noam Chomsky, *Studies on Semantics in Generative Grammar*, Mouton & Co., The Hague, 1972.

[5] Fred J. Damerau, *Problems and Some Solutions in Customization of Natural Language Database Front Ends.*, ACM Transactions on Office Information Systems, April 1985, Vol. 3-2.

[6] Fred J. Damerau, *An Interactive Customization Program for a Natural Language Database Query System*, Cooperative Interfaces to Information Systems, Eds. L.Bolc and M.Jarke, 1986, Springer-Verlag Berlin Heidelberg.

[7] Bipin C. Desai, J. McManus and Philip J. Vincent, *A Portable Natural Language Interface*, AFIPS Conference Proceedings, Vol. 56, 1987.

[8] Bipin C. Desai, J. McManus and Philip J. Vincent, A Natural Language Interface to a Multiple Database Office, Information System SIGOIS Bulletin, Vol. 9-4, December 1988.

[9] Bipin C. Desai, Li Zhang, *Multilevel Interface to a Distributed Database System*, Methodologies for Intelligent Systems, Sixth International Symposium, ISMIS'91,, Eds. Z. W. Ras and M. Zemankova, 1991.

[10] Bipin C. Desai, L. Zhang, *Multilevel Interface to Database Management System: MIDBMS*, Computer Science Report, Concordia University, 1991.

[11] Charles Fillmore, *A Case for Case: Universals in Linguistic Theory*, Holt, Rhinehart and Winston, CBS College Publishing, 1968.

[12] R. P. Gabriel, *Deliberate Writing*, Natural Language Generation Systems, L. Bolc and D.D. MacDonald Eds., Springer-Verlag, New York, 1988.

[13] B. J. Grosz, *TEAM: A Transportable Natural-Language Interface System*, Proceedings Applied Natural Language Conference, 1983, Santa Monica, Ca.

[14] C. D. Hafner, K. Godden, *Portability of Syntax and Semantics in Datalog*, ACM Trans. on Office Information Systems, Vol 3-2, 1985.

[15] E. H. Hovy, *Generating Language With a Phrasal Lexicon*, Natural Language Generation Systems, L. Bolc and D. D. MacDonald Eds., Springer-Verlag, New York, 1988.

[16] Ishikawa et al., *A Knowledge-Based Approach to Design a Portable Natural Language Interface to Database Systems*, Proc. of the 1986 International Conference on Data Engineering, IEEE Computer Society.

[17] P. S. Jacobs, *PHRED: A Generator for Natural Language Interfaces*, Natural Language Generation Systems, L. Bolc and D.D. MacDonald, Eds., Springer-Verlag, New York, 1985.

[18] K. Kukich, *Fluency in Natural Language Reports*, Natural Language Generation Systems, L. Bolc and D. D. MacDonald, Eds., Springer-Verlag, New York, 1988.

[19] George F. Luger, William A. Stubblefield, *Artificial Intelligence and the Design of Expert Systems*, editor Alan Apt, the Benjamin/Cummings Publishing Co., 1989.

[20] John L. Manferdelli, *Natural Language Interfaces: Benefits, Requirements, State of the Art and Applications*, Natural Language Inc., Berkeley, California, 1987.

[21] M. P. Marcus, *A theory of Syntactic Recognition for Natural Language*, The MIT Press, Cambridge, Mass., 1980

[22] K.R. McKeown, *The TEXT System for Natural Language Generation: An Overview* Proceedings of the 20th Annual Conference of the Association of Computational Linguistics, Toronto, 1982.

[23] Igor Mel'cuk, Nikolaj V. Pertsov, *Dependency Syntax: Theory and Practice*, editor Richard Kittredge, John Benjamins Publishing Co., 1987.

[24] Metcer and D.D. MacDonald, *Mumble '86: Design and Implementation*, COINS Technical Report, University of Massachussetts at Amherst, 1987.

[25] Richard J. Pollock, Bipin C. Desai, *The Design and Implementation of a Heterogeneous Database Management System Prototype*, Computer Science Report, Concordia University, 1988.

[26] J. H. Remko, *Natural Language Interface Systems*, Handbook of Human-Computer Interaction, Edited by Martin Helander, North-Holland, 1988

[27] M.Templeton, J.Burger, *Considerationsfor the Development of Natural-Language Interfaces to Database Management Systems*, Cooperative Interfaces to Information Systems, Eds. L Bolc and M.Jarke, 1986, Springer-Verlag Berlin Heidelberg.

[28] B. Thompson and F. Thompson, *Introducing ASK, A Simple Knowledgeable System*, Proc. Conference on Applied Natural Language Processing, 1983.

[29] T. Winograd, *Understanding Natural Language*, Academic Press, New York, 1972.

[30] W.A.Woods, *Augmented Transition Network Grammar*, In S.C.Shapiro (Ed.), Encyclopedia of Artificial Intelligence (vol.1) New York 1987; John Wiley and Sons.

# 'COMPETENCE-SWITCHING' MANAGED BY INTELLIGENT SYSTEMS

**Edeltraud Egger**

**Technical University of Vienna, Argentinierstr. 8/187, A-1040 Vienna**
email: eegger@email.tuwien.ac.at

**Hardy Hanappi**

**Academy of Science, Kegelgasse 27, A-1030 Vienna**
email: awhana@lezvax.oeaw.ac.at

## ABSTRACT

This article deals with problems of time-management and the coordination of activities of heterogeneous agents. To support the planning of strongly interwoven jobs more than a pure scheduling algorithm is needed: Since hierarchical power-relations, diverging objective functions and unequally distributed decision competencies prohibit any straight-forward solution, a more complex framework switching between computer support and face-to-face decisions is more adequate. We discuss these issues in the context of hospital organization, in particular for the example of surgery-planning.

## 1. INTRODUCTION

In human societies work is a group process. Different persons perform different tasks at well-defined time-points to achieve a common goal. Therefore there has to be some kind of co-ordination for these heterog eneous agents[1]. Recent developments in automatization do not focus anymore only on mere machine-co-ordinated production processes, but invade also areas of decision-making and control. This trend has not been paralleled by a theoretical treatment, which could be a reason for the apparent frustration with many of these tools. Since time has been introduced as relation between the single heterogeneous agents within an organisation, it is embedded in decision-making and control with respect to the given power-structure. Due to this a more basic discussion on switching competence between person and machine is urgently needed.

---

[1] [1] distinguish between three needs which determine time-patterns in groups: the need to set and meet deadlines, the need for dynamic teamwork and the need to assure an adequate demand/capability match.

Chapter 1 of this article deals with one approach to solve problems of time-management, namely scheduling-algorithms. Though they are of great importance for many fields of applications, it is evident that they do not cover more complex decision structures of large-scale organizations. In chapter 2 we will show where and why these short-comings appear. This critique leads us to a system-design in chapter 3 which we consider as intelligent and which helps to improve on prevailing solutions. Finally chapter 4 presents an empirical case-study: operation-planning in a surgery-clinic.

## 2. TIME - SCHEDULING

All working-processes consist of a sequence of single tasks, which have to be co-ordinated. In this context *co-ordination* is defined as the order of tasks and their start- and end-times.

e.g.:    C(wp):= (task1, task2, task3, task1, task4, task1)

with $task_1$[t0, t1], ... $task_j$ [tn-1, tn]
with wp: working-process, [t0, t1]: t0 ... start-time, t1 ... end-time

This means that *time* is introduced as *relation* between the agents: It is determined when a certain task has to start, when it has to end and which task has to be finished before another one can begin. It is evident that there is an underlying objective-function which can also be expressed by terms of times: The objective-function describes the expected goal variable, which should reach an optimum, a maximum or a minimum[2] in a certain time-interval. These connections and dependencies of tasks and times can be represented graphically by network-techniques. The according scheduling-algorithm optimized with respect to time computes an optimal solution[3].

Fig. 1. Graph of time-sequence



---
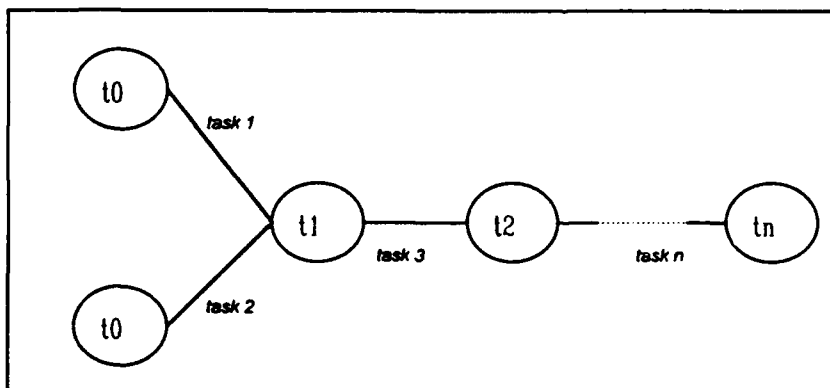
[2] While a maximum is wanted in production, mimimum is searched in questions of cost.

Usually constraints to be met have to be formulated. Temporal constraints can be e.g. the absence of agents at certain times[4], no clear duration of certain tasks, no clear time-points but time-intervals in between tasks etc. Scheduling-algorithms also offer the possibility of priority-setting to arrive at unique solutions. As a consequence it can be defined which tasks are the most important ones.

In literature there are two main approaches to scheduling problems. The traditional one stems from operation-research using methods of optimization like illustrated above[5]. The most practical problem is the enormous solution-space which has to be searched in order to find good solutions. To make the search-process more efficient (in terms of time and memory-space) genetic algorithms[6] are introduced which build on search-techniques like random search, hill climbing and sampling.

The second approach is considering scheduling as knowledge-intensive activity which requires mechanisms for representing and acquiring knowledge[7], both questions of the field of AI. This leads to the constraint-based approach, which is a combination of propagation techniques and specified constraints. Still there is the problem of applicability to real-world-problems, where information is not only uncertain but even incomplete.

The above mentioned ideas of scheduling can also be found in the context of questions of time-scheduling. Usually scheduling problems are under specified concerning organizational characteristics of the problem to be solved. Normally there is a given hierarchical power-structure, where decision-competence is focused on one agent. This agent usually determines the objective-functions and has the power to ignore other agents' objective-functions[8]. Time is treated as quantifiable variable, as a resource-parameter, which enters his objective-function. A typical application-field of this kind of scheduling is a production-process, where machines are programmed to steer the sequence of tasks. Human agents are seen as part of the machinery, they do not have the power to participate in the optimization-process.

In working-areas with a less exclusive and inflexible division of competence this kind of scheduling has to fail[9]. The treatment of time as a measurable scalar does neglect the political character of time. [8] illustrates that successful running of electronic calendars and maintaining them are based on two conditions: First, each of the concerned groups has to invest additional work in updating electronic calendars and second has to leave time-scheduling to a supervising system. Both aspects are not fulfilled in organizations where agents are able to insist on their time-autonomies.

In the following chapter we will discuss other issues arising when seeing automized co-ordination as pure scheduling-problem.

---

[3] compare e.g. [2]

[4] Therefore electronic calendars are needed to describe this kind of temporal constraints.

[5] see also [3]

[6] compare also [4]

[7] For more detailed discussion see e.g. [5], [6].

[8] e.g. In many cases the objective functions of workers are not the same as those of employers.

[9] for further discussion also see [7]

## 3. LIMITS OF TIME - SCHEDULING

Since the agents within a production-unit are heterogeneous, it would be an arbitrary event if they had not conflicting interests. The degree of participation in decision-making is strongly interwoven with the position within the hierarchy. Decision-competence is unequally distributed, even though a certain decision can effect the whole organization. This leads to following problems when trying to establish a scheduling-algorithm:

Fig. 2a. Time-conflicts in organizations: time-autonomies

throughput-related income

high autonomy

fixed working-times

no autonomy

fixed working-times with buffer-times

limited autonomy
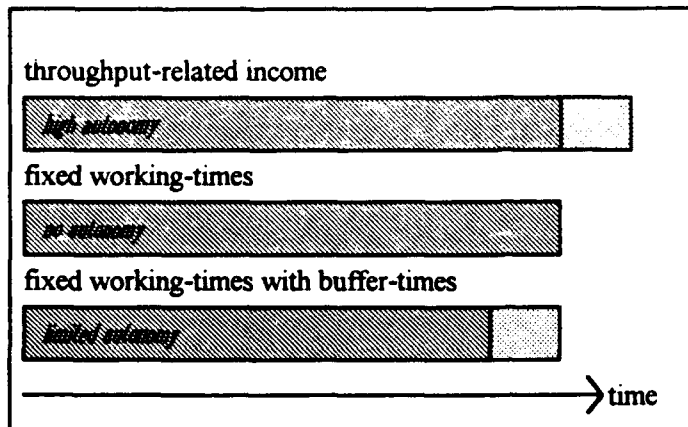
$\longrightarrow$ time

While the group with throughput-related income will use their high time-autonomy to expand their working-times, the group with fixed working-hours is interested in keeping to their working-hours as defined in their working-contracts. The third group, agents with limited autonomy, also will try to stick to their working-hours, but do have a special interest in allocating self-managed buffer-times.

Co-ordination of these types of time-orientation surpasses the capacities of most scheduling-algorithms. External information on power-relations between groups is needed. Either there is one group (e.g. the group sticking to fixed working-hours without autonomy) which determines the final outcome - all the others have to adjust - or there is a weighting of the importance of different groups. Still the questions where these power-relations come from is open.

Fig. 2b. Time-conflicts in organizations: sources of bottle-necks
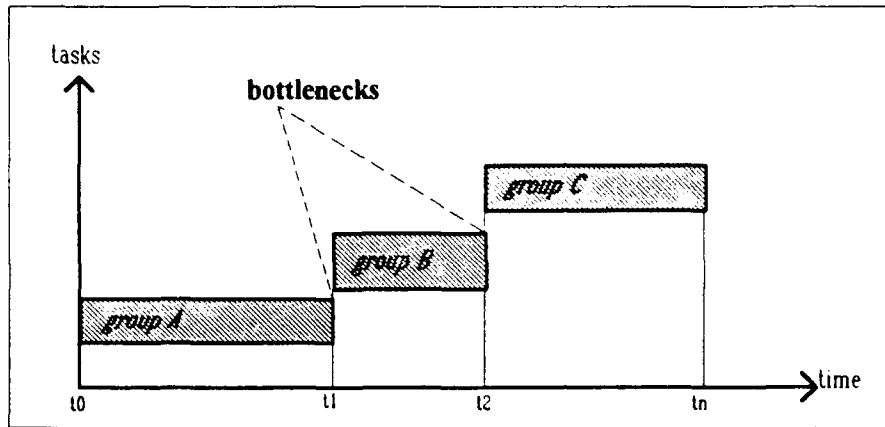


As already mentioned co-ordination means to fix the order of single tasks. The time-point from the end of one task to the beginning of the next one is always a source for bottlenecks. Reasons for the extension of the task of group A are unforeseen complications or unrealistic estimates of task-duration. On the other hand group B could fail to begin with their task at the planned time-point because of lack of time-discipline etc.

Especially the last aspect (time-discipline) is very much connected with an agent's autonomy, which itself is specified by the internal power-distribution.

Problems of bottlenecks are severe in organizations where the different tasks of the single agents are strongly interwoven and depending on each other. Even if some kind of buffer-times is foreseen, the perception of responsibility for bottlenecks can be quite different. The consequences for the originator of bottlenecks depend again from his position within the hierarchy.

Automized time-scheduling can to some extend take organizational factors into account (by using constraints, weighting, etc.), but the designer has to decide which 'view' he/she will implement - which usually is the view of the one who is buying the system - without considering the other agents' wishes. As empirical studies show lack of acceptance of certain decisions can lead to passive counter-reaction. In organizations where decisions are not understandable there is some kind of institutionalized counter-action. It is quite easy to find reasons for delays, slow-down of work and the like. Trying to eliminate obstructions would mean expanding control, which goes hand in hand with increasing cost. This effect does also arise in cases, where certain decisions are made by computer systems. Instead of forcing users to accept automized decisions by extended control it seems to be

more adequate (and cheaper) to further the acceptance of decisions by transferring critical ones to face-to-face-meetings[10].

This means that a general discussion on competence-switching between person and machine has to be started. As we will show in the next chapter this is not a trivial requirement for the design of computer-systems.

## 4. INTELLIGENT COMPETENCE-SWITCHING

Agents in organizations have on the one hand explicit knowledge concerning the power-structure and on the other hand mental maps of informal relations among the members. They also can locate their own position in this structure, though they need not necessarily accept it. Decisions which are conceived as threatening their position are critical ones. Transferring such decisions to a information-system leaves to the concerned agents only institutional ways of defence[11]. This is so because there is no terrain for negotiation. Negotiation in this context means facing conflicting interests and elaborating modes to handle them. Even if there exist underlying contradictions which cannot be eliminated by negotiating there still might be some room for ameliorating the prevailing situation. The most efficient way to negotiate actually is face-to-face.

Let us discuss this issue with respect to the examples given in figures 2a and 2b: If the group with the high autonomy and throughput-related income (compare fig. 2a) is powerful enough to force the other groups to adjust to their choice of time-frame, the weak groups will try to find ways of institutional response. Diverse reactions are open to them: passive defence through low work-intensity, invoking union actions, ...

As soon as a bottleneck appears there is the question who caused it. In organizations with a strong hierarchical structure the manager perceives who is responsible and initiates appropriate sanctions. The person concerned might have a different view and in that case will respond in ways similar to those described in the first example.

These different perceptions of the same bottleneck will also appear in organizations with a less rigid hierarchy. Since in this case there is no possibility of direct sanctioning, the diverse assessment of reasons will lead to passive counter-actions. Both situations can be improved by the introduction of face-to-face negotiations giving the single agents the opportunity to articulate their views.

Obviously face-to-face meetings cannot be *substituted* by automatized decision-making, but information-systems can *support* negotiations by structuring them[12].

---

[10] Critical decisions are the ones which undermine a person's or a group's autonomy or position within the organization.

[11] This kind of reaction usually is counter-productive and leads to high cost.

[12] [9] justifies the increasing demand to computer supported decision-making with 'losses' of productivity in group decision-making occuring because of certain individual's domination, group pressure (which leads to conformity of thoughts) etc.

Therefore intelligent competence-switching modules can be used for demonstrating consequences of certain decisions. But before going into details let us first analyze the meaning of 'intelligence' in this respect.

[10] describes that in complex systems agents are unable to identify all of the constraints on their actions and that their information-processing capacities are limited[13]. The common characteristic of artificial-intelligence techniques is the attempt of modelling decision-making[14]. In our understanding an information system can only be called intelligent if it is able to *simulate* the interaction of *different* agents with *different* world views to further in a direct feedback their co-operation. In this context *co-operation* is not restricted to cases where agents have the same goals, it particularly occurs and needs simulation support in cases of diverging or even opposite objectives of agents[15]. It is this property of simulating contradictions, of anticipating actions and opinions of others leading to these actions, which can be used to solve them or at least to transform them to improve solutions[16].

Fig. 3. Simulating contradicting views
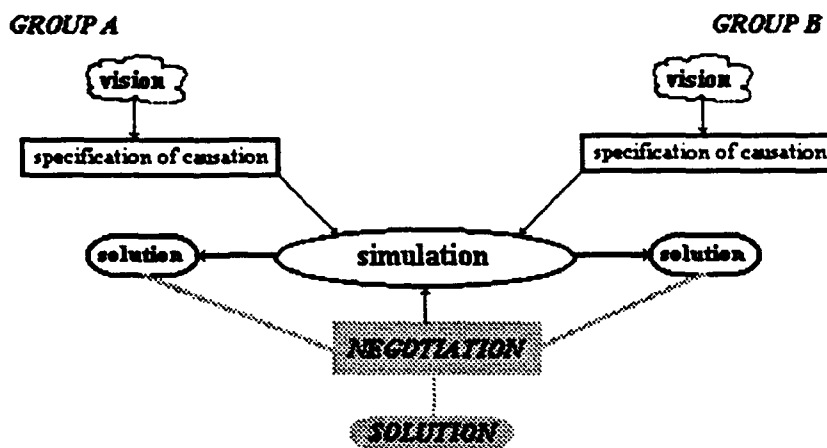


All groups involved in organizations produce more or less simple preliminary 'visions' of internal structures and their own role within the hierarchy. Information systems can be

---

[13] [10, p. 25]

[14] or in other words 'to cast the problem in a frame' [11].

[15] We insist that these contradictions exist in reality and are not just different views which can be eliminated by communication as for example proposed by [12, pp.488-550].

[16] Since contemporary organizations ban all underlying processing contradictions from the internal sphere, they permanently reappear in the outer political field. For an innovative game theoretic treatment of contradictions compare [13].

used to verify these visions by inducing agents to make their objectives explicit. The system-designer provide them with a causation structure (excluding what we called *critical decisions* before) which then is used by simulations[17].

The result of a simulation run are a preliminary solutions which differ from group to group. The choice of a certain solution as final solution - which is a *critical decision* - cannot be delegated to the information-system. At this point competence-switching from the system to face-to-face has to take place. Nevertheless groups enter the bargaining-process more informed since they have preliminary solutions at hand. Assessing and balancing of preliminary solutions is the main purpose of negotiations. A particular agreement can be used as a new input for another simulation - a reverse competence-switching takes place. This recursion is terminated by a final face-to-face agreement[18]. Learning as defining characteristic of intelligent system-design is guaranteed by these recursions. The next chapter will illustrate some of our major issues.

## 5. CASE-STUDY: OPERATION-PLANNING IN A SURGERY CLINIC

The clinic, part of a large university hospital, consists of five departments, representing different specialities[19]. The underlying organizational structure is characterized by a strict hierarchical order and the existence of well-defined disjunctive professional groups (namely surgeons, surgery-nurses, anaesthesiologists). The clinic disposes of six operation-theatres. There is a fixed allocation-plan which assigns operation-theatres to departments (see figure 4).

Fig. 4. Allocation-plan

|  | o.p. 1 | o.p. 2 | o.p. 3 | o.p. 4 | o.p. 5 | o.p. 6 |
|---|---|---|---|---|---|---|
| MON | ward A | ward A | ward B | ward C | ward C/D | ward E |
| TUE | ward A | ward A | ward C | ward E | ward C | ward B |
| WED | ward A | ward A | ward B | ward C | ward C | ward E |
| THU | ward A | ward A | ward B | ward D | ward D | ward E |
| FRI | ward A | ward A | ward C | ward D | ward C/D | ward E |

In other words each ward has certain operation-days. The planning of operations is a several phase-process. Pre-planning is done by entering an operation (including name of patient and responsible surgeon) into a book which is located in the clinic's main secretariat. A pre-program for the following day is discussed in a 14.30 session in which one representative from each department (normally a doctor), the head surgery-nurse and one representative from anaesthesiology are present. In this session the schedule for the

---

[17] Simulations probably will use the scheduling algorithms described in chapter 2 as subprograms.

[18] An indispensable rule for negotiating is to define finite durations (see [14]).

[19] For a more detailed description of the case-study see [15].

next day is set up. The result of these consultations is a time-table which shows the distribution of operations over the available theatres. Nurses decide among themselves (in a 7.00 meeting the following morning) who is going to join which surgical team. No pre-planning for the following days is done, although surgeons may enter operations ahead.

Normally a day's program cannot be realized as it was planned. Due to unrealistic time-estimates, unforeseen complications, emergencies and organizational delays, ad-hoc adjustments have to be made. Responsible for all kinds of re-scheduling is one main surgeon, who, as a consequence of his role, monopolizes valuable information (especially knowledge on vacant operation-capacities) and is only selectively involving implicated colleagues in his decisions.

As can be seen from this brief description all criteria for the emergence of time-conflicts as discussed more generally in the preceding chapters are met: strong hierarchical structure, disjunctive groups (implying different views), different time-autonomies and the urgent need for planning.

Consider again figure 2a: In this case-study the group with high time-autonomy evidently are the surgeons. They press for longer working-days since their income is related to the number of treated patients. The group with no autonomy can be identified as surgery-nurses who want to keep to fixed working-times. Anaesthesiologists - as group with limited autonomy - are interested in self-managed buffer-times to contact next day's patients.

As empirical investigations prove surgeons as the most powerful group set starting times for operations close to the end of the working-day to force nurses and anaesthesiologists to do overtime. As a reaction - we called it *institutionalized counter-action* above - there is passive resistance of surgery-nurses in form of delaying routine work during the day.

What we showed in figure 2b now can be interpreted in the following way. Bottlenecks do occur when e.g. anaesthesiologists (group B) have to wait with the introduction of the anaesthesia for the surgeon (group C). An analogue situation arises between surgery-nurses (group A) and anaesthesiologists.

This last example leads us to the discussion of contradicting views: While surgeons argue that time can be saved by introducing anaesthesia before they arrive and therefore identify anaesthesiologists as the source of the bottleneck, the latter counter-argue that because of surgeons' high time-autonomy there is an uncertainty concerning the arrival of them. Because of medical reasons they do have to wait with the introduction of anaesthesia - and therefore they accuse surgeons to be responsible for bottlenecks.

Simulations making these views and their consequences for planning explicit can be used to structure and support face-to-face-negotiations between the concerned groups. Therefore an intelligent computersystem has to include features managing the treatment of critical decisions. During the design phase it has to be discussed which decisions are not critical and could therefore be solved by implemented algorithms. E.g. how to distribute additional surgery-time: This can be done by the use of certain priority-lists expressing the

hospital's favorite specialities (heart-surgery, ...). Critical decisions on the other hand will be singled out by the computersystem and have to be treated during face-to-face-meetings. E.g the cancelling of operations because of time-constraints, falls under this category, since on one side organizational conditions (working-hours of staff) have to be met, whereas on the other side the operation is very urgent. But this is not the only example of competence-switching in our case-study[20].

There are a lot of other cases where automized decision-making fails. The system-designer has to anticipate critical decisions and therefore implement a "collision-management"-module, which detects them and provides a print-out with such topics for the face-to-face-meeting. In cases where more sophisticated "what-if"-arguments appear in face-to-face-discussions, collision-management can be extended to include simulations of such questions (e.g. the effects of a change of priorities among a hospital's specialities).

## 6. CONCLUSION

We have shown that *competence-switching* between person and machine is an interesting alternative to the increasing unreflected automatization of group decision-making. It will be an important ingredient of the design of intelligent information systems, because it combines elements of organizational development with the more technical aspects of computer support.

## REFERENCES

1. J.E. McGrath, J.R. Kelly, "Time and Human Interaction: Toward a Social Psychology of Time", Guilford Press, New York, 1986.

2. F. Böhm et al., "Mathematische Standardmodelle der Operationsforschung", Verlag Die Wirtschaft, Berlin, 1972.

3. E.L. Lawler et al., "Recent Developments in Determenistic Sequencing and Scheduling: A Survey in Deterministic and Stochastic Scheduling", Reidel, 1982.

4. J.H. Holland, "Adaption in Natural and Artificial Systems", University of Michigan Press, Michigan, 1975.

5. T.J. Grant, "Lessons for OR from AI: A Scheduling Case Study", Journal of the Operational Research Society, 37/1, 41-57, 1986.

---

[20] For a more extended discussion of this issue see [16].

6.  M.S. Fox., N. Sadeh, "Why is Scheduling difficult? A CSP Perspective", in: Proc. ECAI-90, 754-767, 1990.

7.  J. Grudin, "Why Groupware Applications Fail: Problems in Design and Evaluation", Office: Technolgy and People 4:3:245-264, 1989.

8.  S.F. Ehrlich, "Social and Psychological Factors Influencing the Design of Office Communication Systems", Proc. CHI+GI´87 Human Factors in Computing Systems, Toromto, 1987.

9.  G.P. Huber, "Group Decision Support Systems as Aids in the Use of Structured Group Management Techniques", DDS-82 Conf. Proc. 1982: 96-108, 1982.

10. S. Moss, J. Rae, "Artificial Intelligence and Economic Analysis", Edward Elgar, London, 1992.

11. G. Hanappi, "Evolutionary Economics", Habilitation at the Technical University of Vienna, Department of Economics, 1992.

12. N. Luhmann, "Soziale Systeme", Suhrkamp, Frankfurt a.M., 1984.

13. S. Brams, "Negotiation Games", Routledge, New York, 1990.

14. A. Strauss, "Negotiations", Jossey-Bass, San Francisco, 1979.

15. E. Egger, "Gruppenentscheidungssysteme am Beispiel der Terminplanung", Technical Report 6/91, Technical University of Vienna, Department of CSCW, 1991.

16. E. Egger, I. Wagner, "Time-management: A Case for CSCW?", Proc. of CSCW-Conference, ACM Press, Toronto, 1992.

# STRATEGY ACQUISITION BY AN ARTIFICIAL NEURAL NETWORK: EXPERIMENTS IN LEARNING TO PLAY A STOCHASTIC GAME

Neal M. Mazur

Department of Electrical Engineering and Computer Science
Union College
Schenectady, NY 12308, U.S.A.

## ABSTRACT

Artificial neural networks have been successfully used to perform a variety of tasks, mostly in the areas of pattern recognition and classification. This paper explores their use in learning and executing strategies. A series of experiments are performed in which a network using the backpropagation algorithm learns strategies for playing the game of casino blackjack. The experiments range from the supervised learning of a blackjack strategy using the normal backpropagation regime, to supervised learning using situations generated from the naturally occurring probabilities of the game, and, finally, to an unsupervised experiment where the network learns a strategy based on its performance and past experience in playing. The latter experiments introduce probabilistic and contradictory feedback to the network due to the stochastic nature of the game. The experiments show that artificial neural networks can be used to represent and learn strategies for a stochastic game and that their performance can be favorably compared to that of human players.

## 1. INTRODUCTION

Artificial neural networks in general, and the backpropagation algorithm[1] in particular, have been shown to be powerful pattern recognition tools[2,3,4]. This paper will explore the use of the backpropagation algorithm in representing and learning strategies associated with the game of casino blackjack. Casino blackjack is first adapted for use with backpropagation networks and then a series of learning experiments are performed.

The first experiments utilize the normal backpropagation mechanism of supervised learning. A set of patterns and desired outputs, as prescribed by an expert, are continually cycled through the network during training. Sejnowski and Tesauro have had outstanding success in teaching backpropagation networks to play backgammon using this approach[5,6].

The second set of experiments again models an expert strategy but utilizes a passive observation mode. In this case, the network views an expert

working in the naturally occurring environment of the game. The network can no longer request to see a particular situation but must instead deal with the differing probabilities of situation occurrence.

The final experiment does not involve an expert. The network learns a strategy based on its own performance during play of the game. This introduces problems associated with credit assignment and contradictory feedback to the system based on the stochastic nature of the game. The ability to learn based on performance has a long history in artificial intelligence from Samuel's landmark work with the game of checkers[7] to more recent work utilizing evolutionary replicators[8] and stochastic automata[9]. The backpropagation algorithm will be utilized here with a reward or penalty[10] given to actions based on the outcome of each hand.


## 2. CASINO BLACKJACK

Casino blackjack is a gambling game played with a standard deck (or several decks) of playing cards. A player competes against a dealer who utilizes a fixed strategy. The object of the game is to accumulate a greater hand value than the dealer without going over 21. The cards two through ten are evaluated at face value, jacks, queens and kings have a value of 10 and aces are given a value of 1 or 11 based on which value will favor the holder. A hand that contains an ace counted as 11 is termed a **soft** hand. All others will be called **hard** hands.

In the casino game, there may be several players playing against a single dealer. Each player makes a bet for the hand. The initial play sees two cards dealt to each player and the dealer with the last of the dealer's cards exposed (the **show card**). Each player, in turn, plays out his hand. The player may choose to **stand** on his given hand, **hit** (request another card) or **double down** (double the bet to receive only one card).

After receiving an extra card with a hit, the player may continue making hit decisions. After an initial hit decision, the play ends when the player makes a stand decision or when the held cards total more than 21 (called a **bust**, in which case the player immediately loses).

After all players have completed making their decisions, the dealer completes the **hand** by continually taking extra cards until the cards that are held total 16 or greater. If the value is more than 21, the dealer loses to all players who did not bust. Dealer values from 17 to 21 beat players with lower hand values, tie players with equal hand values and lose to players with greater hand values.

There are three decision variables that a player considers when making decisions. The player's hand total is obviously a major factor in decision making. A total of greater than 11 introduces the possibility of busting when drawing a card while low totals will be beaten by a dealer hand that does not bust. A value of 11 is favorable for doubling down. The second decision variable is whether the player's hand is soft or hard. A soft hand cannot go over 21 with the drawing of a card. The value of the dealer's up card is the last decision variable. Values of 4 through 6 indicate a stronger possibility that the dealer will bust than when a 10 is shown.

One could also devise a fourth decision variable and use a history of the cards that have been played to influence betting amounts and decisions. Card counting strategies are considered illegal at the gaming tables and are combatted by using a large number of decks (from 4 to 6) and reshuffling cards often. This paper will not consider this variable.

There are various postulated optimum blackjack strategies in the literature[11-14]. All such strategies considered are either the same or differ by a small number of decisions. We will not be concerned with the slight variations and will adopt the soft and hard strategies from Silberstang shown in Figures 1 and 2.
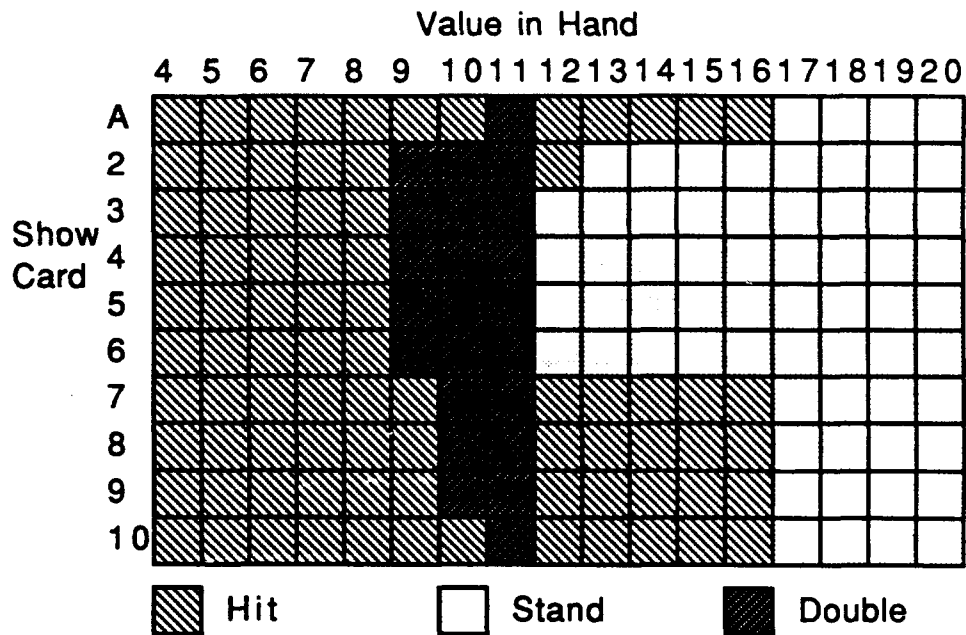


Fig. 1. Silberstang strategy for hard blackjack hands.

## 3. THE BACKPROPAGATION ALGORITHM

The standard backpropagation algorithm utilizes a network consisting of layers of non-symbolic nodes to learn to recognize a set of patterns. The learning is supervised as each time a pattern is presented to the network, the desired output for the pattern is also provided. The network gradually converges to a point where all patterns are properly associated with the correct output. These networks are general purpose pattern recognizers and have desirable generalization and robustness properties.

**Value in Hand**
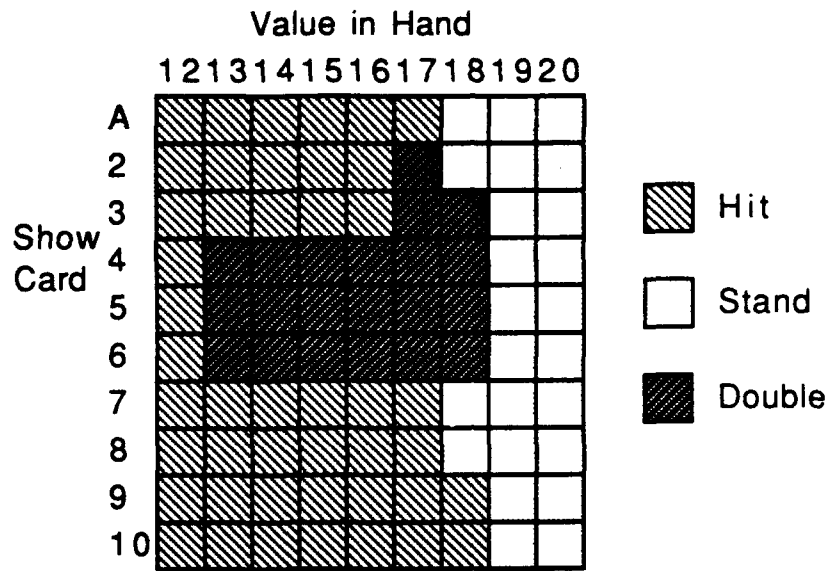
Fig. 2. Silberstang strategy for soft blackjack hands.

The network is maximally connected from layer to layer. The connection between each pair of nodes has a weight associated with it. The weights are given initial random weights which are adjusted as the network learns.

Each pattern is presented to the network, in turn, during training. A presentation can be separated into three steps: propagation of activity, computation of error and weight adjustment.

The pattern is encoded as a binary number and the activity of each node of the input layer is assigned one bit of the number. This input layer activity is propagated forward through the layers of the network using the formula:

$$a_j = \frac{1}{1 + e^{-\Sigma( a_i \cdot w_{ij} )}}$$

where $a_j$ is the activity of the receiving node, the $a_i$'s are the activities of the nodes in the sending layer and $w_{ij}$ is the weight on the connection between nodes i and j. This formula generates an output from the network at the final layer. The propagation of activity can be used exclusively to generate a response from a given input in a non-learning mode.

After propagation, an error term is computed for each non-input node in the network. The error associated with each output node is given by

$$e_i = a_i(1 - a_i)(a_i - d_i)$$

where $d_i$ is the desired output for node i. The output node errors are propagated back to interior level nodes

$$e_i = a_i(1 - a_i) \cdot \sum_k (e^k \cdot w_{ik})$$

Lastly, the weights are adjusted by the following term such that another presentation of the given pattern will yield outputs closer to those desired

$$\Delta w_{ij}(n) = \eta(e_j \cdot a_{ik}) + \psi \cdot \Delta w_{ij}(n)$$

where $\eta$ is a constant that controls the speed of learning. $\Delta w_{ij(n)}$ is the change in weight on the $n^{th}$ iteration of training, and $\psi$ is a momentum constant. The momentum term inhibits forgetting of patterns that have been previously presented.

Normally, this presentation of patterns and adjustment of weights continues until the outputs derived from propagation of all patterns are within some threshold, $\tau$, of the desired outputs.

## 4. APPLICATION OF THE NETWORK TO BLACKJACK

All possible situations and responses in the game of casino blackjack must be represented by the network. An input node will be created for each possible value of each decision variable. This creates 17 nodes for the possible player hand values, 4 through 20, 10 nodes for the possible show cards (ace through 10) and 2 nodes to represent the status of the player hand as being soft or hard. There are 260 possible inputs that may be presented to the network. Hands with hard values make up 170 of these scenarios and hands with soft values make up the other 90 (since there are no soft hands with values from 4 through 11). Similarly, there are 3 output nodes, one for each possible decision of hit, stand and double. Figure 3 shows the architecture of the network used in the experiments to follow. As described above, there are 29 input layer nodes and 3 output layer nodes that were dictated by the problem. There will also be one interior layer that consists of 40 hidden nodes.

All programming experiments that follow were written in C and run on a DECstation 5000 Model 200 running ULTRIX V4.2.

## 4.1 AN EXPERIMENT IN SUPERVISED LEARNING

The first experiment to be performed has the network learning the Silberstang strategy using the traditional backpropagation technique. All possible input patterns are cycled during training in a given order. The learning is supervised since the network is supplied with the desired output for each input pattern. The network is doing active experimentation in the sense that it is requesting the strategy to give its responses for specified

situations. The outputs supplied are consistent and correct as compared to the given strategy.
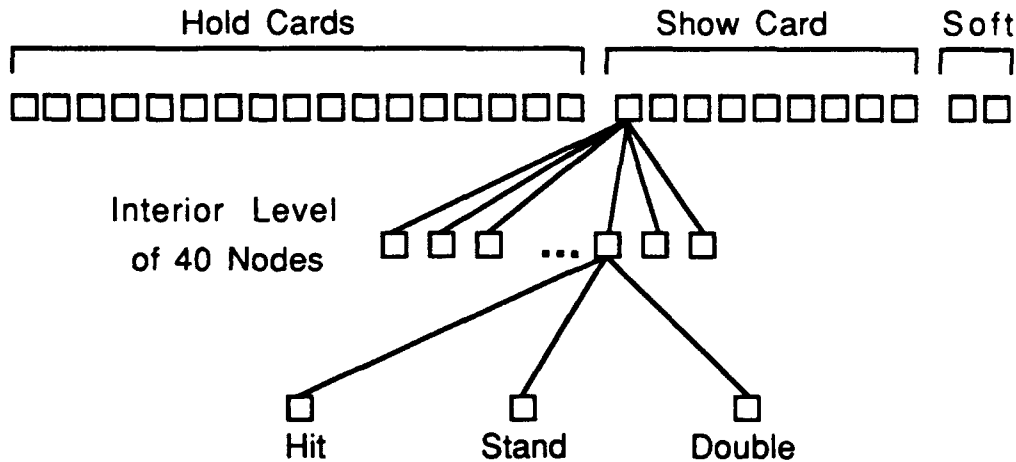


Fig. 3. A neural network architecture representing a casino blackjack strategy.

The algorithm was trained 100 times with different initial random weight assignments in the range (0.0 ... 1.0). The speed of learning constant, $\eta$, and the momentum constant, $\psi$, from the weight change formula were set to values of 0.2 and 0.8, respectively. The value of the convergence threshold, $\tau$, was set at 0.1. A training epoch is defined as one presentation of each of the 260 input patterns to the network. The results of the experiments are summarized in Figure 4.

| $\eta$ = 0.2 | $\psi$= 0.8 | $\tau$ = 0.1 | 260 input patterns | | |
|---|---|---|---|---|---|
| network architecture : 29 -> 40 -> 3 | | | | | |
| values below in epochs for 100 trials | | | | | |
| mean | median | min | max | std dev | fails |
| 91.96 | 89 | 72 | 173 | 15.78 | 0 |

Fig. 4. Results of experiments training an artificial neural network on an expert casino blackjack strategy using the standard backpropagation regime.

The experiments show that the backpropagation algorithm was highly successful in learning the given strategy. The 260 patterns were learned in all tests and the number of epochs needed for training was small. These results are not surprising given the fact that strategy acquisition problem has been translated into one of binary pattern recognition. The

backpropagation algorithm has been shown to work well for problems of this type.

## 4.2 SUPERVISED LEARNING IN A NATURAL ENVIRONMENT

The next experiment puts the network into a passive mode of operation while learning the strategy. Instead of following the typical pattern of repeatedly cycling through all possible inputs, the network will view a sequence of blackjack nands that would occur during the normal play of the game. The training will still use the backpropagation algorithm and will essentially be supervised. It is analogous to the network watching a player who utilizes the Silberstang strategy play the game at a casino blackjack table. The network will once again see a consistent, correct strategy.

The test for convergence to the optimum strategy will be changed for practical reasons. The actual outputs will not be within a certain threshold $\tau$ for all patterns. Some patterns will occur much less frequently (for instance, those representing soft hands and small hard totals like 4) due to the probabilistic nature of the game. The difference in frequency of presentation of patterns to the network makes reaching a specified tolerance for all patterns difficult. Convergence to the given strategy will be said to be reached when the output node representing the desired action (hit, stand or double) has a greater activity than the other output nodes for every pattern. That is, if majority rule was used, the correct action would be chosen in all cases.

There were 10 tests run in this passive mode of operation with different initial random weight assignments in the range (0.0 ... 1.0). The results of the experiments are shown in Figure 5.

| $\eta = 0.2$ | $\psi = 0.8$ | majority convergence | | | |
|---|---|---|---|---|---|
| network architecture : 29 -> 40 -> 3 | | | | | |
| values below in hands played for 10 trials | | | | | |
| mean | median | min | max | std dev | fails |
| 28137 | 20655 | 11820 | 83180 | 22373 | 0 |

Fig. 5. Results of experiments training a backpropagation neural network on an expert casino blackjack strategy using a sequence of normally occurring hands.

All tests reached majority convergence in less than 100,000 hands of play. These results are quite encouraging. Even though the number of hands seems quite large, the speed of learning compares favorably to the results in the active experimentation mode. Each epoch represents 260 decisions which roughly translates into 130 hands using the assumption that two decisions are made each hand. The convergence was thus reached after

the equivalent of approximately 90 epoches for the minimum trial and 640 epoches for the maximum trial.

Additionally, the network learns a large portion of the strategy very quickly. A majority of the iterations needed to reach convergence are associated with correcting the last few erroneous decisions. Figure 6 illustrates this for a typical run from the experiments. It graphs the number of incorrect decisions, those that do not agree with the strategy being modeled, versus the number of hands viewed. The number of incorrect decisions was computed every 100 hands played. The data points did not all monotonically decrease as shown in the graph. Only the values that showed decreases in the number of incorrect decisions were shown to give a general idea of the trend in improvement. Note that once a strategy did reach 0 incorrect decisions, it never degraded to more than 1 incorrect decision and after several hundred extra iterations would move back to and remain at 0. It can be seen from the graph that all but 5 decisions were learned in the first 4100 iterations. The final three quarters of the hands viewed merely corrected the last 2 percent of the decisions.

## Incorrect Decisions vs. Hands Viewed
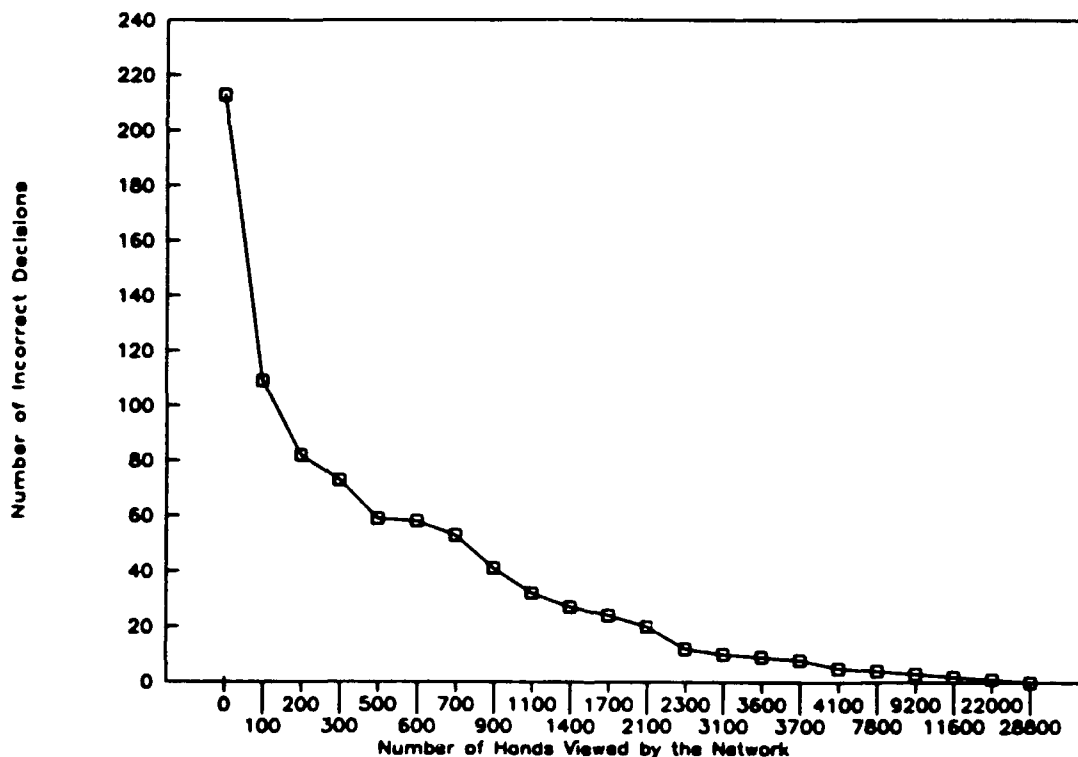


Fig. 6. Graph showing the decrease in the number of incorrect decisions prescribed by the backpropagation network as it passively observes an optimum strategy.

One of the major advantages of artificial neural networks is that they more closely mimic human learning than more traditional techniques. This experiment illustrates this nicely. The network learns much as a human would in that it observes a strategy working in the normal environment, quickly learns the basics of the strategy, displays forgetting (as stated above, at times the performance degrades) and finally learns the entire strategy (note that it may outperform, in this case, because a human may never master the decisions for all situations).

## 4.3 UNSUPERVISED LEARNING BY PERFORMANCE EVALUATION

The final experiment is the most challenging. The network will learn to play the game without observing an expert but by playing the sequence of games itself. The network will not be supervised in the standard sense. It will be informed whether it won a hand or not but will not be informed of the proper decision(s) for the hand.

Thus, the network will sometimes be given incorrect information and will often be given inconsistent feedback as compared with information that it has previously received. The information can be incorrect in several ways. First, the network will be told that it lost a hand when in fact the decisions that it made were correct. Since the game of casino blackjack is ultimately a losing one for the player, this will happen often. Conversely, there is the possibility that an incorrect decision will lead to a winning hand. Finally, if the network makes several decisions during a hand, some may be correct and some may be incorrect and we have a good example of the credit assignment problem. Regardless of the outcome in this case, the feedback will be in some way erroneous. The feedback will be inconsistent because the network will sometimes be told that a decision in a certain situation is correct and at other times told it is incorrect. The hope in running the experiment is that the network will filter through the noise to generate an appropriate strategy.

The game will be slightly simplified for this experiment by disallowing the double action. This removes the need for an extra variable that would be necessary to indicate the number of cards in the player's hand (since doubling down can only be performed when two cards are held) and will accommodate the feedback rule which will be used.

The backpropagation network and learning algorithm are once again used. The network will be the same as in Figure 3 except that only two output nodes, hit and stand, are now necessary. Decisions will be rewarded and penalized very simply. If the network wins or has a draw for a given hand, each decision made during the playing of the hand will be presented to the network for one training cycle. If the network loses a hand, the opposite of each decision made during the playing of the hand will be presented to the network for one training cycle. This is analogous to a person playing the game without being told the rules. The only information provided is that there is a choice of two actions and after a hand that the player has won or lost.

Three experiments were performed with different initial weights in the network, varied in the range (-1.0 ... 1.0). Each experiment consisted of 1,000,000 hands. The number of incorrect actions for all 260 decisions when

compared to the Silberstang strategy was recorded every 1000 hands.    Figure 7 displays these results.

| $\eta = 0.2$ | $\psi = 0.8$ | convergence not reached | | |
|---|---|---|---|---|
| network architecture : 29 -> 40 -> 2 | | | | |
| values below in incorrect decisions (minimum followed by iteration number) | | | | |
| | min (it#) | max | mean | median |
| Trial 1 | 6 (970000) | 160 | 41.85 | 39 |
| Trial 2 | 12 (870000) | 166 | 39.50 | 36 |
| Trial 3 | 9 (947000) | 150 | 39.05 | 36 |

Fig. 7.    Results of experiments training a backpropagation network to play blackjack using a normal sequence of hands and a performance based reward system.

The strategies generated were quite volatile.    Even though they reached their minimum values for incorrect decisions late in training, the strategies did not stabilize at these low values.    The strategies did not at any point mimic the optimal strategy with no incorrect decisions (although they were only compared every 1000 hands and could have matched at a point between tests).

However, there were some promising results.    Although there was a great variance from test point to test point, averaging the number of incorrect decisions over 50 test points shows a gradual improvement.    Figure 8 shows this graphically for values from Trial 1.    The average number of incorrect decisions approaches a value of about 34 for all three trials.

Next, consider the strategies that are generated after hand 1,000,000. The number of incorrect decisions for the three trials at this point were 11, 45 and 17, respectively.    Although there were a number of incorrect decisions in each strategy, many of them were on the borderline of changing.

The incorrect decisions came in two categories.    The first is associated with situations that occur infrequently (such as a hard 4).    Several occurrences of incorrectly rewarded decisions, due to chance, in recent history can incorrectly swing the action for these situations.    The second category is that of statistically tough decisions.    The question of whether to hit a hard 15 or 16 given a show card of 8, 9 or 10 is not clear cut.    In general these are losing hands for the player.    The difference between the activation of the chosen action and the opposing action when a situation is presented to the network gives a rough level of confidence in the decision.    A portion of the decision space for Trial 1 is given in Figure 9.    The results show that the network has discovered to some extent which of the choices are obvious and which are difficult.

## Average Incorrect vs. Hands Viewed
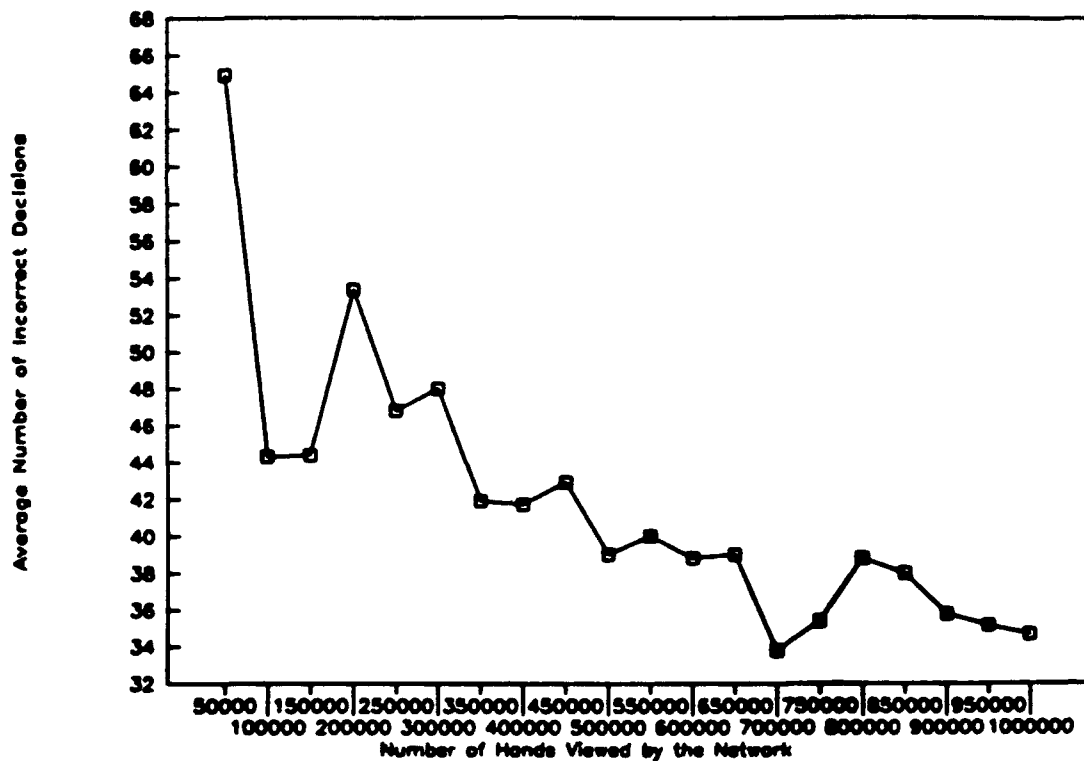


Fig. 8. Graph showing the average number of incorrect decisions prescribed by the performance supervised backpropagation network every 50,000 hands.

### Show Card

| | | 8 | | 9 | | 10 | |
|---|---|---|---|---|---|---|---|
| Value in Hand | 15 | 0.0007 | hit | 0.0015 | hit | 0.0002 | hit |
| | 16 | 0.0003 | hit | 0.0011 | hit | 0.0001 | hit |
| | 17 | 0.011 | stand | 0.0063 | stand | 0.0016 | stand |
| | 18 | 0.57 | stand | 0.0046 | stand | 0.001 | stand |
| | 19 | 0.85 | stand | 0.61 | stand | 0.058 | stand |
| | 20 | 0.89 | stand | 0.91 | stand | 0.8 | stand |

Fig. 9. A portion of the blackjack strategy learned by the network includes a coarse level of confidence along with the decisions.

One can try to remove the variance in strategy quality by developing several strategies and melding them into a meta-strategy. For example, the

strategies in the three trials can be combined by taking the majority choice for each decision. The strategy in Figures 10 and 11 is obtained.
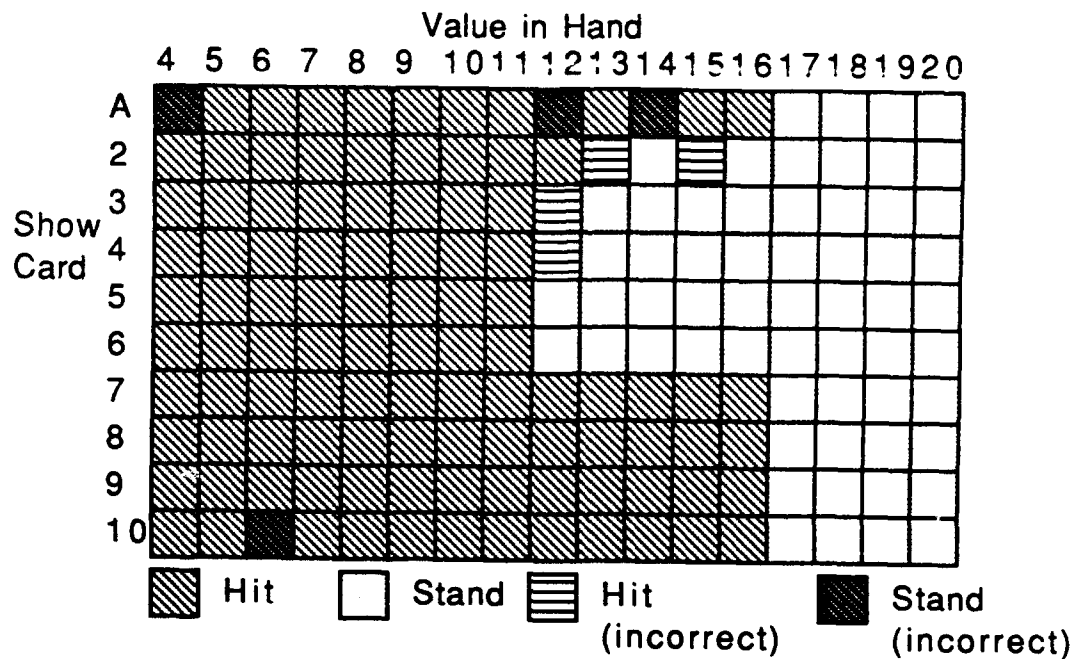


Fig. 10. Hard hand meta-strategy with differences from Silberstang strategy indicated.
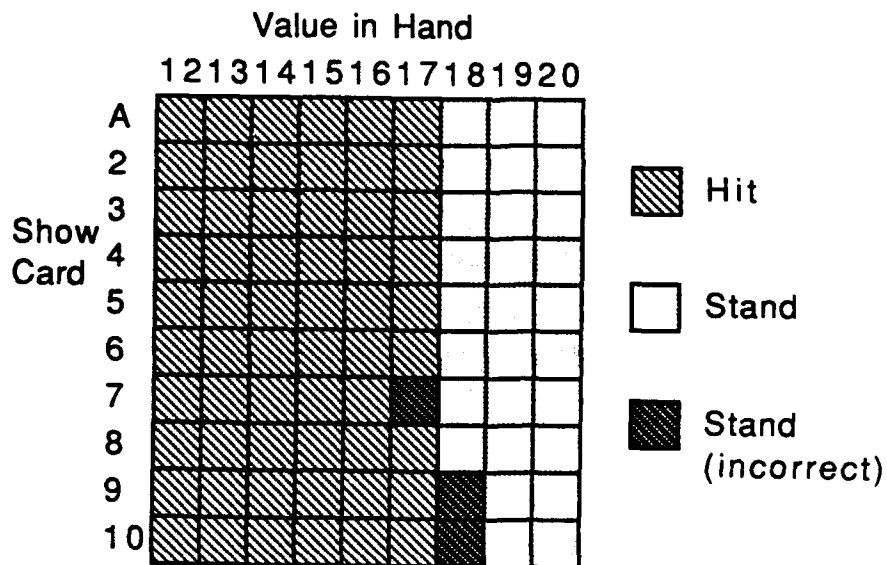


Fig. 11. Soft hand meta-strategy with differences from Silberstang strategy indicated.

There are 11 discrepancies with the hit and stand strategy from Silberstang. Although there are 11 incorrect choices by majority, at least one of the trial strategies had the correct action for every decision except one (a soft hand with show card 7 and value in hand 17).

Finally, the performance of the learned strategies will be considered. They will be compared to the Silberstang strategy, two intuitive strategies, no bust and dealer, and a random strategy. A no bust strategy always hits hard hands below 12 and stands on all other hard hands. It hits on all softs hands below 17 and stands on all others. The dealer strategy uses the same hitting and standing rules as the dealer. Table 1 shows the results for 1,000,000 hands played with each comparison strategy and the meta-strategy. Only the last 500,000 hands of the trial strategies are used (recorded while learning) to discount early random behavior.

**Table 1. Comparison of Blackjack Strategies**

|          | Wins   | Losses | Draws  | Loss % |
|----------|--------|--------|--------|--------|
| Silber   | 432805 | 474980 | 92215  | 4.22   |
| No Bust  | 421962 | 508392 | 69646  | 8.64   |
| Dealer   | 408659 | 488934 | 102407 | 8.03   |
| Random   | 319552 | 628487 | 51961  | 30.89  |
| Trial 1  | 209202 | 249859 | 40939  | 8.13   |
| Trial 2  | 208383 | 250784 | 40833  | 8.48   |
| Trial 3  | 210593 | 248661 | 40746  | 7.61   |
| Meta     | 427908 | 481224 | 90868  | 5.33   |

The last column in the table gives the difference between the number of losses and wins as a percentage of the number of hands played. The best strategies will minimize this percentage. The strategies created in the trial runs were of the same order as the intuitive strategies. The meta-strategy was an improvement over all but the Silberstang strategy which has slightly better performance.

## 5. CONCLUSIONS

A series of experiments has demonstrated that a backpropagation neural network can be adapted for use in representing strategies for the game of casino blackjack. The network can learn an expert strategy quickly and without failures when allowed to work in a mode of active experimentation. This is the normal mode of operation for backpropagation training.

The network is also able to model an expert strategy when working in a passive mode of operation. That is, it learns the strategy by observing an expert who is presented with a normal sequence of hands. In doing so, the network exhibits many characteristics of human learning such as fast learning of the basics of the strategy, occasional forgetting of portions of the strategy and a lengthy period of fine tuning. The network eventually mastered the observed strategy in all tests.

Lastly, the network developed a strategy not by observing an expert but by learning from its own experience in playing. It rewarded actions that led to victories and penalized those which led to defeats. This type of learning introduced contradictory feedback to the system due to the stochastic nature of the game. The strategies created in this manner had roughly the same level of performance as some intuitive human strategies. A meta-strategy built from the end products of several trials approached the performance level of an expert strategy.

## REFERENCES

1. D. E. Rumelhart, G. E. Hinton, R. J. Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing*, vol. 1, MIT Press (1986) pp. 318-362.

2. D. J. Burr, "Experiments with a Connectionist Text Reader," *Proceedings of the IEEE First International Conference on Neural Networks*, 1987, pp. 717-724.

3. G. L. Martin, J. A. Pittman, "Recognizing Hand-Printed Letters using Backpropagation Learning," *Neural Computing*, no. 3, 1991, pp. 258-267.

4. R. H. Silverman, A. S. Noetzel,"Image Processing and Pattern Recognition in Ultrasonograms by Backpropagation," *Neural Networks*, no. 3, 1990, pp. 593-603.

5. T. J. Sejnowski, G. Tesauro, "A Parallel Network that Learns to Play Backgammon," *Artificial Intelligence*, vol. 39, no. 3, 1989, pp. 357-390.

6. G. Tesauro, "Neurogammon Wins Computer Olympiad," *Neural Computation*, vol. 1, no. 3, 1989, pp. 321-323.

7. A. L. Samuel, Some Studies in Machine Learning using the Game of Checkers" in *Computers and Thought*, eds. E. A. Feigenbaum, J. Feldman, McGraw-Hill, 1963.

8. P. Kurka, "Evolution of Replicators Playing a Strategic Game," *Biological Cybernetics*, vol. 52, no. 4, 1985, pp. 211-218.

9. K. Okamura, T. Kanaoka, T. Okada, S. Tomita, "Learning Behavior of Variable Structure Stochastic Automata in a Three Person Zero-sum Game," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC 14, no. 6, 1984, pp. 924-931.

10. S. Lakshmivarahan, K. S. Narendra, "Learning Algorithms for Two-person Zero-sum Stochastic Games with Incomplete Information: a Unified Approach," *SIAM Journal on Control and Optimization*, vol. 20, no. 4, 1982, pp. 541-552.

11. E. Silberstang, *How to Gamble and Win*, Franklin Watts, Inc., 1977.

12. R. A. Epstein, *The Theory of Gambling and Statistical Logic*, Academic Press, 1977.

13. D. Ortiz, *On Casino Gambling*, Dodd, Mead & Co., 1986.

14. H. Tamburin, *Casino Gambling, The New Complete Guide on How to Play and Win*, Research Services Unlimited, 1988.

# VIEWPOINTS AND SELECTIVE INHERITANCE IN OBJECT-ORIENTED MODELING

**Markku Oivo**

**Technical Research Centre of Finland (VTT)**
**Computer Technology Laboratory**
**P.O. Box 201, SF-90571 Oulu, Finland**
**email: markku.oivo@vtt.fi**

## ABSTRACT

Object-oriented modeling is a powerful and natural knowledge representation mechanism for many real-life applications. However, there are situations when the traditional object-oriented modeling and inheritance lattices are not sufficient. We introduce a dynamic viewpoint mechanism with selective inheritance for building models in the context of software engineering and for viewing the models from multiple perspectives. Furthermore, we describe a set of inter-object relationships which can be used to enhance the traditional object-oriented and frame-based modeling mechanisms. These methods and techniques have been implemented in a prototype system and they have been used to model various software engineering activities and elements.

## 1. INTRODUCTION

The way we perceive concrete or abstract objects depends on our viewpoint. A chemist, a broker on the Amsterdam oil spot market, a car driver and an automobile engine all have a different viewpoint to oil. They all see some common properties of oil but they also attach very different properties to the same object. Having all the attributes attached to the same object all the time would be unnecessary and confusing for any of them. Similarly, objects in software engineering can be viewed from different viewpoints. A software module may have different attributes when viewed from developer's, tester's, manager's or customer's viewpoint. Certainly there are, and there should be many common attributes as well, but having all the attributes visible for everyone would make the object overwhelmingly complicated and difficult to comprehend. Hence, we need a mechanism for looking at objects from different viewpoints.

Object-oriented modeling is a powerful and natural knowledge representation mechanism for many real-life applications. It yields itself naturally into taxonomic classification and the inheritance mechanism makes it easy to incrementally define features for objects. However, there are situations when the traditional object-oriented modeling and inheritance lattices are not sufficient.

We may want to create an object as an instance of a class and then at some point continue developing both the ancestor and the descendant independently, i.e., changes in the parent class should not be inherited into the child anymore. Consider the task of creating and maintaining reusable components in software engineering. An object may have been initially created as a part of a class hierarchy. When we store it into a reuse repository we may not want to store the whole chain of its ancestors in the class hierarchy. Instead, we could take all the inherited attributes with the object and store the objects as a self sufficient element into the repository. Later the original class hierarchy may be changed and classes may be modified but we may want to keep the original version of the object in the repository.

The above mentioned examples require an object-oriented system to support:

- consistent internal representation of the objects
- traditional object oriented modeling with both single and multiple inheritance
- selective inheritance of attributes from multiple sources
- dynamic changing of viewpoints and consequently attributes and behavior of an object based the user's current interest

In this paper we propose a viewpoint mechanism with selective inheritance which will address these problems. It is possible to implement these mechanisms on top of many of the object-oriented as well as frame-based systems but a truly efficient system requires these features to be implemented in the underlying system. Our method does not replace or exclude neither single nor multiple inheritance. It is an addition to the traditional object-oriented and frame-based classification and inheritance mechanisms. We have implemented a prototype system (ES-TAME) which fulfills the key features of the viewpoint mechanism with selective inheritance. ES-TAME is built in PC environment using Kappa, ToolBook and Excel tools in Windows environment. It has been developed and used in the context of top down goal oriented characterization of software engineering activities. We are using reusable and tailorable object-oriented models to represent software engineering elements. Our methods are especially useful in the analysis and design activities of software development.

## 2. MODELING

Naturally, the basic object-oriented modeling methods and Is-A hierarchies are used extensively when building models of software engineering artifacts. In addition, we provide a set of predefined inter-object relationships and a dynamic viewpoint mechanism with a highly selective inheritance for building various model hierarchies and networks. By offering a limited collection of relationships we can maintain consistent models and provide automated support for managing the models. On the other hand, if we would simply use attributes to store relationships without a rigorous set of rules we could easily end up to a spaghetti-like relationship network which is very difficult to maintain in a large modeling application. With a well-defined set of relationships we can build models which are flexible and yet manageable. In this paper we concentrate on the dynamic viewpoints and selective inheritance and we will describe the inter-object relationships only briefly in this paper. An extensive discussion of the inter-object relationships can be found in [12].

## 2.1. DYNAMIC HIERARCHIES

The viewpoint mechanism is based on dynamic hierarchies concept, which is realized with the dynamic manipulation of inheritance hierarchies. Basically, our *Is-A / Children* and *Instance-Of / Instances* relationships are similar to the standard class/subclass and class/instance relationship offered by most object-oriented and frame-based systems [3], [4], [5], [11], [8], [18]. They are the only relationships which employ the conventional inheritance in ES-TAME. However, we do not currently provide traditional multiple inheritance. Instead, we provide dynamic linking of the Is-A relationships which considerably enhances the capabilities of the traditional inheritance. Each object can have a potential Is-A relationship, or more precisely inheritance link, to several super classes but only one of them is active at any point in time. All the attributes of the active super class are inherited, whereas inheritance via the other inheritance links is highly selective and must be explicitly defined. The purpose of the children relationship is to catalogue all the subclasses or instances of a given class.

Despite the rather novel approach to inheritance we still consider our system a class based system as opposed to prototypical and object based systems. In prototypical systems the traditional inheritance is replaced by linking a new object to a prototype object. A message can be delegated to the prototype if the new object has an identical response to that particular message. On the other hand, the behavior of the prototype can be redefined or new features can be added simply by defining a method in the new object. Our approach can offer a similar functionality as delegation on attribute level but normally we do not use it on class or object level. An object may use the method or attribute of another object but normally we do not use a full object as a prototype, we simply inherit selected attributes from one or more classes. We always have a parent class (which can be changed dynamically) from which we inherit all the methods and attributes. Additional functionalities and attributes can be inherited selectively from other objects.

A well known object based system Self has the notion of prototype metaphor instead of classes and variables [19]. It searches values for slots using parent pointers instead of inheriting according to a class hierarchy. Self does not use classes or variables. In ES-TAME each class or instance always has an active parent class and inherits all the attributes of that parent. If we did not have the dynamic linking mechanism, our strategy could be considered a single inheritance approach similar to what is used in systems like Smalltalk, KEE, and Eiffel. However, the dynamic linking mechanism of the inheritance links provides multiple viewpoints to object models. Furthermore, it facilitates context sensitive behavior for objects by changing relationships on the fly and inheriting new attributes and functionalities from the new parent. The old relationships can be restored without any loss of information due to the dynamic relationship manipulation.

The fact that we do not currently use multiple inheritance does not mean that we would argue that it is useless in the context of software modeling and construction. On the contrary, it is easy to identify numerous cases where objects are conceptually related to more than one parent and multiple inheritance is useful. The multiple viewpoints and selective inheritance offer many of the benefits of multiple inheritance. We can avoid the well known name collision and repeated inheritance problems involved in multiple inheritance [3], [17], [20] because we have only one parent link or

viewpoint active at any point of time. Consequently, we can avoid the problem of having several possible methods in multiple parent classes with the same name when forwarding a message to upwards in the inheritance lattice. However, we would still gain from having both the multiple inheritance and the dynamic viewpoints with selective inheritance in our toolbox.

During a link change, all the application level local values of an object, i.e. instance values which are not inherited from the old parent, must be maintained in the object in order to be accessible also under the new parent. Furthermore, all the attributes selected by the user to be inherited and ported under the new parent must also be maintained. We can recover these values if the old parent becomes the current parent again. Even if the old parent is deleted or is otherwise not accessible anymore, we can maintain the attributes which were initially inherited from the deleted parent before the link change. This is useful for reusing objects in new systems where the parent may not be included in the new systems. Attributes without a local value and which are not explicitly defined to be maintained by the user can be removed in the object level. If the inheritance link is changed to point back to the old parent the attributes are automatically inherited again from the old parent. Consequently, there is no need to maintain these attributes while the inheritance link points to a new parent.

This mechanism avoids the problem of information maintenance involved in coercion in schema evolution for object-oriented databases [15]. In schema evolution the coercion mechanism discards information during type changes if their definition is not included in the new type. In the dynamic linking of the inheritance link ES-TAME maintains all the information even if the definition of the attribute is not included in the new parent. On the other hand, dynamic linking of the inheritance link is most often used as a run-time feature and the relationship can also be changed back to any of the previous parents, as opposed to the one way evolution of versions in object-oriented databases [15]. The following algorithm describes the principle of the attribute manipulation of an object Object during dynamic changing of an Is-A link from Old-Parent to New-Parent (figure 1).

```
FOR EACH attribute inherited from the Old-Parent in the Object
      IF attribute has a local value in the Object
      OR attribute is selected by the user to be inherited THEN
         Make attribute local in Object and maintain
         the local values
      ELSE
         Remove attribute from Object
Change IS-A link of Object to the New-Parent
```

Fig. 1. Principle of attribute manipulation.

## 2.2. RELATIONSHIP NETWORKS

In addition to the basic Is-A relationship we need a set of pre-defined relationships, which can be used as basis for the model building tools and graphical browsers in a similar manner as the basic Is-A relationship. We have defined all the relationships in pairs because of the emphasis of using ES-TAME to build reusable objects. Each object can be taken out of its original hierarchy and subsequently be stored into the reuse repository for future use. It must retain knowledge not only of its descendants in the hierarchy but also of its possible ancestors, parts if it is a composite object, to which context it belongs and information on how its relationships can be used in new applications. It is a reusable object with relationships as connectors which can plug into other objects both upwards and downwards in any of the relationship hierarchies.

The relationships offered by ES-TAME are Is-A/Children, Instance-Of/Instances, Part-Of/Has-Parts, Compatible-Objects, Dynamic-Attribute and a Counterpart relationship:

- The *Part-Of / Has-Parts* relationship pair is used to describe compound objects. A composite object is a collection of objects which can be managed as a single entity. Our composite object is roughly comparable to the related concepts of some other object-oriented languages and object-oriented database systems [9], [17]. However, it is important to notice that we do not require a composite object to be instantiated in a top down fashion starting from the compound object and then instantiating the components [1], [17]. Due to the emphasis on reusable components and parallel design in large projects, we don't have any restrictions on the order in which compound objects are built and instantiated.

- The *Compatible-Objects* relationship is used to describe objects which can be used together, e.g. the function point method might be compatible with MIS projects but not with real-time projects.

- The *Counterpart relationships* are provided for creating various domain specific relationships and links between objects. They are normally used to define relationships between objects which are used in the same context to build a larger scheme. The counterpart relationship has some similarities with Booch's association relationship which denotes a semantic connection among otherwise unrelated classes [3]. Using counterpart relationships the user can create, edit and browse any kind of application specific hierarchies. Naturally, each object can also be viewed from all the standard viewpoints provided by ES-TAME. These relationships are used to manage the interconnections and interactions between the related objects, including message passing, constraint reasoning and value propagation.

- The *Dynamic-Attribute* provides a way of associating an object's attribute with the attribute of another object [9]; e.g. if we have estimated the number of source lines (SLOC) in the product characterization and given it as an attribute to the product model, we can link the corresponding SLOC attributes of the resource estimation and defect slippage models to the product model's SLOC attribute. Thus we maintain the SLOC estimate in one place only and changing the estimate can be automatically updated in the other models.

# 3. USING DYNAMIC VIEWPOINTS AND SELECTIVE INHERITANCE

We provide a mechanism for attaching a generic viewpoint mechanism to any of the models or model components and their relationships. The objects and models are always internally defined by a single internal representation. Normally each user has a default viewpoint to the system. For example, the tester is mainly interested in the errors, faults, and various quality models. On the other hand, management is more interested in budgets, resources, cost, project schedule, etc., and can have models tailored according to the management perspective. The manager may impose a schedule for the whole project using the project model and estimations of the size and effort needed to implement the system. When the system reaches the testing phase the tester takes a totally different viewpoint to the software. The tester performs the pre-defined testing procedures and records the results. The information recorded in the model of the software is available also for the tester.
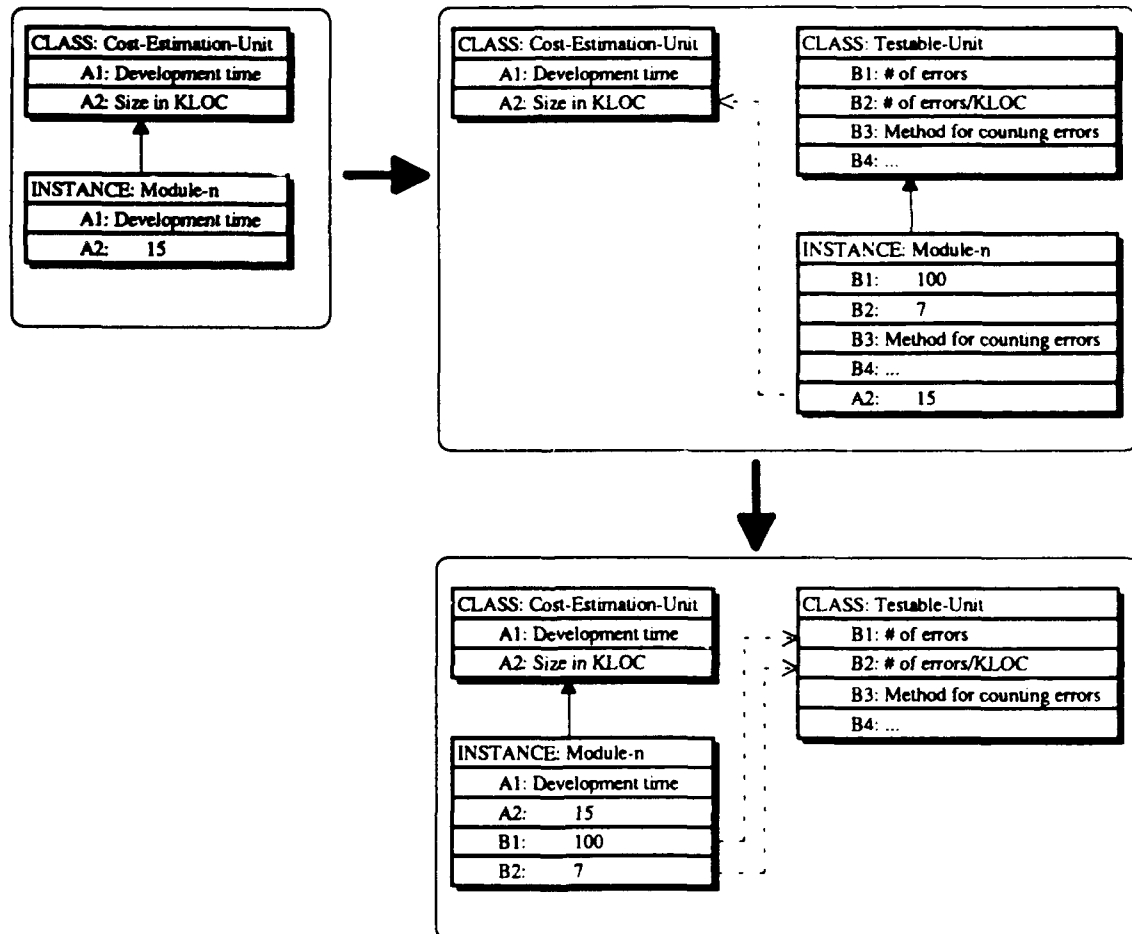
**Fig. 2. Internal structure of objects during changing viewpoints.**

Each model or component of a model is defined as an object. Each object is defined with attributes which are relevant to itself as a class or as an instance of a class. For example, a data flow diagram is defined with its relevant attributes in the context of structured analysis and design. However, because it is defined as a model which can be viewed from multiple perspectives it has the capability of having several viewpoints. If the user wants to examine the quality aspects of a particular data flow diagram, he/she would change the viewpoint of that object to a particular quality model. As a result, the data flow diagram would be dynamically linked to that quality model and inherit its features and functionality. Note that this is different from multiple inheritance. Linking is dynamic and inheritance is applied only while the object is linked to the viewpoint. When changing the viewpoint again, only those attributes which are instantiated during the old viewpoint, i.e. those that have been modified or given local values, are ported into the new viewpoint.

```
CLASS: SystemViewpoints
      CLASS: Testing
      CLASS: ResourceEstimation
            CLASS: Static
                  CLASS: FunctionPoint
                  CLASS: SLOC
                        CLASS: COCOMO
                              CLASS: Basic
                              CLASS: Intermediate
                                    CLASS: Advanced
                        CLASS: IBM.FSD
                        CLASS: Doty
                        CLASS: BaileyBasili
            CLASS: Dynamic
      CLASS: QualityModels
            CLASS: Cohesion
            CLASS: Coupling
            CLASS: DefectModels
                  CLASS: Defects
                        CLASS: Faults
                        CLASS: Failures
                        CLASS: Errors
                  CLASS: DefectViewpoint
```

Fig. 3. A sample collection of potential viewpoints of a software sub-system.

One of the advantages of the dynamic viewpoint mechanism and selective inheritance is it limits the amount of information in each object. Because most of the objects can be viewed from a variety of predefined perspectives (quality models, cost estimation, testing, design, implementation etc.), use of straightforward multiple inheritance or implementing the attributes and functions as part of the objects would yield excessive information and obscure the user's understanding of the object itself and its conceptual rel tionships to other objects. With dynamic viewpoints we can focus our attention on the features which are relevant to our current interest.

Our approach differs from the multiple interfaces defined in some object-oriented languages. For example, Snyder proposes two different interfaces to classes, one for

public use and one for subclasses [16]. Others have proposed restricted subsets of operations for different users to facilitate multiple views to the same object [7]. Dynamic viewpoints and selective inheritance are primarily a means for changing run-time behavior, object attributes, or even class hierarchies. Changing a viewpoint adds new methods and attributes to an object and may remove old ones if they are no longer needed. This is a basic difference from controlling visibility in Trellis/Owl [14] or accessing an object in [7] and [16].

## 4. CONCLUSIONS

We have presented a method for enhancing the traditional object-oriented techniques to better support the modeling of software engineering related artifacts and activities. For example, we should have different perspectives to a software module when viewed from developer's, tester's, manager's or customer's viewpoint. In addition to the basic object-oriented techniques we provide a dynamic viewpoint mechanism with selective inheritance and a set of predefined inter-object relationships. These techniques have been demonstrated in a prototype system which has been used to model various software engineering elements. The knowledge representation mechanisms have proven to offer better modeling tools for the user. The dynamic viewpoint mechanism offers a convenient way of having multiple perspectives to the models. It is also a useful tool for incrementally building object-oriented models.

## ACKNOWLEDGMENTS

## REFERENCES

1. Banerjee, J. , Chou, H.T., Garza, J.F., Kim, W., Woelk, D., Ballou, N., Data Model Issues for Object-oriented Applications. ACM Transactions on Office Information Systems, January 1987.

2. Basili, V.R. & Rombach, H.D. The TAME Project: Towards Improvement-Oriented Software Environments. IEEE Transactions on Software Engineering, Volume SE-14, No. 6, June 1988, pp. 758-773.

3. Booch, G., Object-Oriented Design with Applications. Benjamin/Cummings Publishing Company, Redwood City, CA, 1991, 580 p.

4. Fikes, R., Kehler, T. , The Role of Frame-Based Representation in Reasoning. Communications of the ACM, Vol. 28, No. 9, September 1985, pp. 904 - 920.

5. Goldberg, A. , Robson, D., Smalltalk-80: The Language and its Implementation, Reading, Massachusetts, Addison-Wesley Publishing Company, 1983

6.  Hahn, U., Jarke, M., Rose, T. Teamwork Support in a Knowledge-Based Information Systems Environment. IEEE Transactions on Software Engineering. No 17, May 1991, pp. 467-482.

7.  Hailpern, B., Ossher, H., Extending Objects to Support Multiple Interfaces and Access Control. IEEE Transactions on Software Engineering. Vol. 16, No 11, November 1990, pp. 1247-1257.

8.  IntelliCorp, KEE Software Development System User's Manual.

9.  IntelliCorp, Kappa System User's Manual.

10. Kim, W., Banerjee, J. , Chou, H.T., Garza, J.F., Woelk, Composite Object Support in an Object-Oriented Database System. Proceedings of the ACM Conference on OOPSLA, October 1987, pp. 118-125.

11. Lieberman, H., Using Prototypical Objects to Implement Shared Behavior in Object Oriented Systems. Proceedings of the ACM Conference on OOPSLA. September 1986, pp. 214-223..

12. Meyer, B., Object-oriented Software Construction. Prentice Hall, New York, 1988.

13. Oivo, M. & Basili, V.R. Representing Software Engineering Models: The TAME Goal Oriented Approach. IEEE Transactions on Software Engineering, Volume 18, No. 10, October 1992.

14. Oivo, M., Multiple Viewpoints to Software Models. International Workshop on Experimental Software Engineering, Dagstuhl Castle, Germany, September 1992 (to appear in Springer-Verlag Lecture Notes Series).

15. Schaffert, C., Cooper, T., Bullis, B., Kilian, M., Wilpot, C., An Introduction to Trellis/Owl. Proceedings of the ACM Conference on OOPSLA, September 1986.

16. Skarra, A.H., Zdonik, S.B, The Management of Changing Types in an Object-Oriented Database. Proceedings of the ACM Conference on OOPSLA, September 1986, pp. 483-495.

17. Snyder, A.., Encapsulation and Inheritance in Object-Oriented Programming Languages. Proceedings of the ACM Conference on OOPSLA, September 1986, pp. 38-45.

18. Stefik, M., Bobrow, D., Object-Oriented Programming: Themes and Variations. AI Magazine, Volume 6, No 4, Winter 1986, pp. 40-62.

19. Stroustrup, B.: The C++ Programming Language, Addison Wesley, Reading, Massachusetts, 1986.

20. Ungar, D., Smith, R., SELF: The Power of Simplicity, Proceedings of the ACM Conference on OOPSLA, October 1987, pp. 227-241, Orlando, FL, 1987.

21. Wegner, P., Concepts and Paradigms of Object-Oriented Programming, Expansion of Oct 4 OOPSLA-89 Keynote Talk, OOPS Messenger, Vol 1, Number 1, August 1990, pp.7-87.

# MULTIVARIATE DISCRETIZATION OF CONTINUOUS ATTRIBUTES FOR MACHINE LEARNING

**Thomas W. Rauber, Dinu Colţuc\* and Adolfo S. Steiger-Garção**

**Universidade Nova de Lisboa–Faculdade de Ciências e Tecnologia**
**Departamento de Informática–Intelligent Robotics Group**
**2825 Monte de Caparica–Portugal**
**Tel.: +351-1-2953220 — Fax: +351-1-2955641 — E-mail: tr@fct.unl.pt**

## ABSTRACT

Symbolic Machine Learning algorithms like decision tree or rule induction programs must discretize continuous attributes. This symbolization of continuous values is usually done in an univariate way. That means that the attributes are transformed from continuous to discrete space independently from each other. For each attribute the infinite range of its possible values is discretized to a finite number of values which from there onwards are considered as symbolic.

In this paper we propose multivariate discretization of continuous attributes as a preprocessing step of symbolic induction. The generated symbolic values represent not only a single continuous attribute, but a multidimensional discretization of many continuous attributes. The novel approach is to merge several continuous attributes in order to discretize them *and* to use this multivariate discretization as a preprocessor for the symbolization of continuous attributes in inductive learning.

Furthermore we propose the $Q^*$ algorithm which is capable of learning multivariate prototypes in an Euclidean space. It is a self-organizing supervised learning method which condenses the raw sample data to a representative set of prototypes. The algorithm can be used for the multivariate quantization philosophy that is proposed in this text.

Experimental results are presented for a classification task. Instead of the normally univariate quantization, multivariate quantization using $Q^*$ is employed.

## 1. INTRODUCTION

In inductive learning the type of an attribute can be continuous, discrete ordered or totally unordered (symbolic, nominal, categorical). Difficulties arise in the generation of decision structures when the objects are described by a *mixed* set of attributes. Consider the example set of Table 1. We modified the original training set for decision tree induction that Quinlan [2] used to illustrate the effect of his ID3 [1]. Originally 14 samples characterize 2 weather classes. *Outlook, temperature, humidity* and *windy* were the four

---

\* ICPE, Research Institute for Electrotechnology, Bucharest, Romania

symbolic attributes that described the classes *positive (P)* and *negative (N)*. We substituted the symbolic values of the two attributes *temperature* and *humidity* by continuous values to obtain a mixed attribute set. It was assumed that the temperature is *cool* between 5°C and 15°C, mild between 15°C and 25°C and hot between 25°C and 35°C. Humidity is *normal* below 80% and *high* above 80%. The original symbolic values are indicated in parenthesis.

What are the alternatives for the attribute model in the mixed attribute case if a classifier has to be induced?

## 1.1 SYMBOLIC → NUMERIC

Table 1: Example training set for inductive learning

| Sample | ATTRIBUTES | | | | Class |
|--------|---------|-------------|----------|-------|-------|
|        | Outlook | Temperature | Humidity | Windy | |
| 1 | sunny | 33.2 (hot) | 89.9 (high) | false | N |
| 2 | sunny | 31.1 (hot) | 92.2 (high) | true | N |
| 3 | overcast | 28.3 (hot) | 98.2 (high) | false | P |
| 4 | rain | 19.9 (mild) | 88.1 (high) | false | P |
| 5 | rain | 10.1 (cool) | 71.2 (normal) | false | P |
| 6 | rain | 6.3 (cool) | 63.2 (normal) | true | N |
| 7 | overcast | 8.4 (cool) | 75.2 (normal) | true | P |
| 8 | sunny | 18.2 (mild) | 92.3 (high) | false | N |
| 9 | sunny | 9.3 (cool) | 78.2 (normal) | false | P |
| 10 | rain | 19.3 (mild) | 69.3 (normal) | false | P |
| 11 | sunny | 21.4 (mild) | 70.4 (normal) | true | P |
| 12 | overcast | 22.4 (mild) | 94.2 (high) | true | P |
| 13 | overcast | 32.2 (hot) | 73.2 (normal) | false | P |
| 14 | rain | 21.8 (mild) | 82.5 (high) | true | N |

The symbolic attributes (outlook, windy) are *encoded* to numeric attributes. Then a purely numeric learning task is performed. This could be for instance a supervised clustering in a multidimensional Euclidean space, a connectionist (neural network) approach, a Bayesian classifier in a multidimensional continuous space etc. The problem with this technique is the eventual creation of a non-existing order that is imposed on the values of the unordered variable, see e.g. Utgoff and Brodley [3] for a discussion on encoding symbolic attributes. In the *weather* example the *windy* attribute can still be handled relatively easy, since it has only a binary range. The value *true* could be encoded to 0.5, *false* to -0.5 and even an *unknown* value finds a place in this model with 0.0. The *outlook* attribute has more than two possible values. One method to encode multi-value categorical attributes is the introduction of new attributes, one for each possible value, and then proceed like for the binary case. Here we would obtain three new binary attributes out of one multi-value discrete attribute: *outlook_overcast, outlook_rain* and *outlook_sunny.* One drawback of this method is obvious. Attributes with many values cause an enormous multiplication of the size of the attribute set.

Many induction programs have to go the way from symbolic to numeric attributes, e.g. all those which rely on a geometric relationship among the attribute values. Kohonen's Learning Vector Quantization [4], [5], Utgoff and Brodley's PT2 decision tree induction [3] or multilayer perceptrons are examples of such learning programs.

## 1.2 NUMERIC → SYMBOLIC

Symbolic induction algorithms that work on symbolic attributes are for instance Quinlan's ID3 decision tree induction algorithm [1] from which the ASSITANT program for medical diagnosis of Kononenko et al. was derived [6]. Michalski's AQ-Family which was successfully applied with AQ11 [7] for soybean disease diagnosis is another example of symbolic induction. Generalization of the data is expressed as rule sets. The CN2 algorithm of Clark and Niblett [8] uses a hybrid form for the decision structure. The so called *decision lists* are a intermediate concept between trees and rules.

The numeric attributes (temperature, humidity) are *quantized*. In Quinlan's original sample set this was certainly done in order to obtain the symbolic values. The infinite range of continuous attributes is discretized to a finite set of symbolic values. Usually, also the still existing order relation of the discrete values is discarded during the induction process. This is the case for ID3.

The discretization of the variables is done however in an *univariate* manner. *Temperature* is discretized without taking into account *humidity* and vice versa. An independence of the class conditional probability distributions for the attributes is assumed:

$$p((temp, humid)| Class) = p(temp| Class) p(humid| Class)$$

or generally for attribute values $x_i$ and classes $C$.

$$p((x_1, ..., x_n)| C) = \prod_{i=1}^{n} p(x_i| C) \tag{1}$$

The assumption of the independence of the attribute values facilitates the induction process in general. The quality of an attribute can be expressed without respecting the correlation that it eventually has to another attribute. Univariate quantization was investigated by Catlett [9]. His discretizing algorithm calculates cutting thresholds "for each attribute without reference to the others". We aim at taking into consideration the *inter*dependence of the attributes when a discretization has to be done, not *in*dependence.

## 2. MULTIVARIATE DISCRETIZATION

### 2.1 GENERAL IDEA

Instead of discretizing attributes univariately one by one, we merge several continuous attributes to a multidimensional vector and discretize in the resulting Euclidean space. The symbolic values are then geometric bounded entities, formed like hyperspheres, hyperrectangles, hyperpolygons, open halfspaces etc.

Let us join the two attributes *temperature* and *humidity* to a multidimensional *superattribute*. The resulting 2-dimensional points are plotted in Fig. 1.

The univariate discretization consists of cutting the continuous range in 3 respectively 2 intervals which are orthogonal to the axes for *temperature* respectively *humidity*.
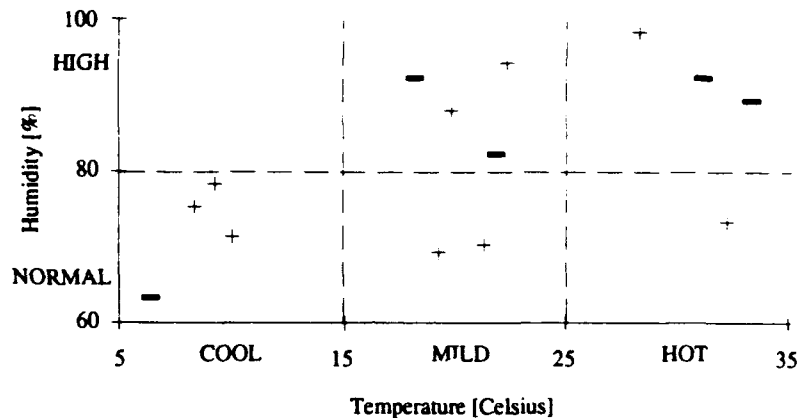


Fig. 1. Several continuous attribute values are points in a multidimensional Euclidean space.

A multivariate discretization would try to create two-dimensional patches which describe the classes. In that sense a rectangle ( ( $15 \leq temp \leq 25$ ) $\wedge$ ( $60 \leq hum \leq 80$ ) ) would be a bivariate symbolization of the two continuous attributes. Any 2-dimensional continuous value which a priori in the training and a posteriori in the classification falls in that rectangle e.g. ($16.9^\circ$C, 78.5%) has the symbolic value of the rectangle. But also a hypersphere (a circle in the 2-dimensional case) around the point ($9.27^\circ$C, 74.87%) is a symbolic value.

Suppose now that some multivariate cluster algorithm had generated cluster centers like in Fig. 2. The *positive* class is represented by three prototypes *P-1* at ($9.27^\circ$C, 74.87%), *P-2* at ($26.28^\circ$C, 71.52%) and *P-3* at ($32.15^\circ$C, 91.05%), the *negative* class is represented by two prototypes, *N-1* at ($6.3^\circ$C, 63.2%) and *N-2* at ($18.2^\circ$C, 92.3%).
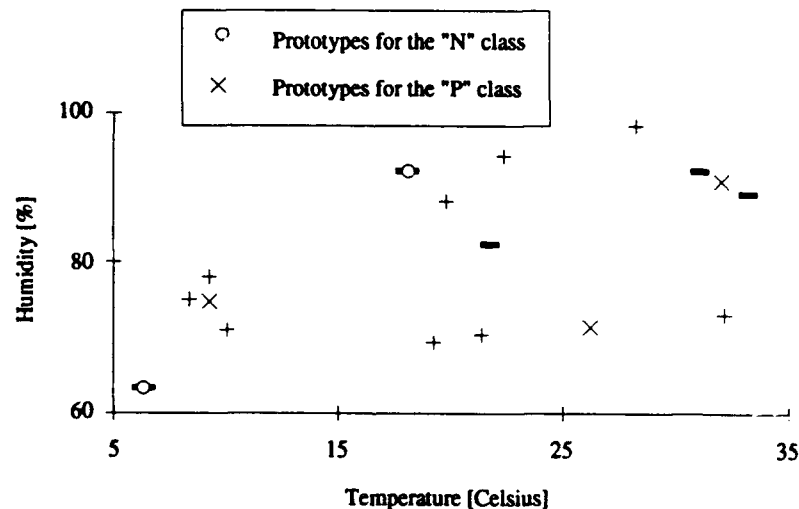


Fig. 2. Prototypes that represent bivariate symbolization of continuous attributes.

We consider these prototypes as the multivariate symbolic values for several continu-
cus attributes. Any unknown multidimensionai vaiue has the symbolic value if the proto-
:ype is nearer to it than any other prototype in terms of a distance function. The bivariate
symbolization of the two attributes *temperature* and *humidity* is now done as follows: For
each of the 14 samples search the prototype which is most similar (next to) to the sample;
then exchange the bivariate continuous value by the symbolic value of the prototype.
From now onwards there are only symbolic values that describe the classes. Proceed with
symbolic induction as usual, e. g. generate rules or decision trees. In the ID3 decision tree
induction algorithm the *gain* for the superattribute could be calculated for instance since it
is a symbolic attribute.

For our example we obtain then a purely symbolic description like in Table 2. The
bivariate continuous values for *temperature* and *humidity* have been substituted by the
symbolic values of the superattribute.

But not only prototype-based multidimensional learning of continuous attributes fits
into this philosophy, but also other geometric shapes that define a multidimensional sym-
bolic value, like hyperrectangles.

Table 2: Multivariate symbolization of the continuous attributes *temperature* and *humidity*

| Sample | ATTRIBUTES | | | Class |
|---|---|---|---|---|
| | Outlook | Windy | Superattribute (Temperature, Humidity) | |
| 1 | sunny | false | "N-3" | N |
| 2 | sunny | true | "N-3" | N |
| 3 | overcast | false | "N-3" | P |
| 4 | rain | false | "N-2" | P |
| 5 | rain | false | "P-1" | P |
| 6 | rain | true | "N-1" | N |
| 7 | overcast | true | "P-1" | P |
| 8 | sunny | false | "N-2" | N |
| 9 | sunny | false | "P-1" | P |
| 10 | rain | false | "P-2" | P |
| 11 | sunny | true | "P-2" | P |
| 12 | overcast | true | "N-2" | P |
| 13 | overcast | false | "P-2" | P |
| 14 | rain | true | "N-2" | N |

## 2.2 BENEFITS AND DISADVANTAGES OF MULTIVARIATE DISCRETIZATION

What benefits can be expected from the merging of several numeric attributes during
discretization? The general tendency in Machine Learning seems to corroborate the need
for multivariate instead of univariate approaches. Decision trees that test on several
attributes are proposed as a more sophisticated evolution of decision trees that test only on
a single attribute. Utgoff and Brodley [3] illustrate the limitations of univariate splits in
decision trees with an example of a linear function that has to be learned. The univariate

splits are only optimal if a projection of an attribute on its respective coordinate axis preserves the class separation. Their PT2 decision tree induction algorithm uses multivariate tests in a single tree node. Also Clark and Niblett's CN2 [8] employs a multivariate test with the concept of the *decision lists*, although the quantization of continuous attributes is done in the usual univariate way. With multivariate tests, decision trees become eventually less complex because class boundaries may not be described exactly by an univariate approach, only approximations to the class concepts could be done. If hyperrectangles are used for instance to define the classes and the probability distributions of the classes really have rectangular forms, a multivariate description is surely preferential.

As a drawback of multivariate discretization we can mention several facts: In a classification system with a high degree of human interaction its becomes more difficult to understand a multivariate test. Decisions made by the system become less transparent. A human being can easily trace the decision making of a tree classifier if only one attribute is allowed in a node, but it gets harder to understand the combination of several attributes in a single step.

## 3. RELATED RELEVANT WORK

In this section we will point out research that has been done in order to solve problems that are related in many different aspects to our approach of multivariate discretization of continuous attributes. We do not claim any completeness of this overview. The objective is to relate the principles that are used in the different learning algorithms to our general approach of multivariate discretization.

### 3.1 MULTIVARIATE CLUSTERING AS DISCRIMINANT ANALYSIS

Rendell investigated the problem of selective induction in a general framework that highlights the expression of *discriminant analysis* [10]. Class formation and multivariate quantization are basically put on the same level. Three different types for symbolic values of multidimensional ordered numerical attributes are mentioned:

i.) Hyperrectangular class clusters. This form of class formation expresses a concept or class as a bounded multidimensional box in which the members of the class are stored. The symbolization process in this case would state e.g. that an example has the symbolic value "box_1" if its first attribute may vary in the interval [3,6], its second attribute may vary in the interval [1,5] etc.

ii.) Open hyperspaces. A line with the equation $y = x$ in the 2-dimensional Euclidean space is an example of a boundary between two different classes. The symbolic value of a sample becomes then "upper halfspace" or "lower halfspace".

iii.) Prototype based methods. This class representation schema is exactly the same as for the hyperrectangles, differing only in the *shape* of the discriminating boundary. A prototype is the centroid of a hypersphere. The symbolic value is associated with the sphere. A sample is labelled with this symbolic value if its multidimensional numerical value falls e.g. into "sphere_1". The $Q^*$ uses a prototype-based model.

## 3.2 HYPERRECTANGLES

Salzberg [11] developed the idea of rectangular shaped class delimiters to a learning algorithm that is able to model multimodal, nested class distributions. His NGE-algorithm dynamically learns by reshaping existing rectangles or creating new rectangles. The learning is supervised and the outcome of a classification during training is the source of feedback for the learning mechanism. The nearest hyperrectangle learning algorithm uses the way from symbolic to numeric attributes. Symbolic attributes are encoded like describ d in section 1.1. Experiments in two medical domains and for Fisher's iris flowers data set [12] are described.

The $Q^*$-algorithm that will be presented later shares some concepts with the hyperrectangle learning. A feedback about how the actual state of the induction system performs is used to dynamically reorganize the prototype based class representation. Differently however $Q^*$ is conceived only as a preprocessing step for numerical attributes in order to transform them to symbolic values. The proper induction is performed by a symbolic approach, like ID3 for instance, whereas the NGE approach works in a geometric space.

## 3.3 SELF-ORGANIZING MAPS, LEARNING VECTOR QUANTIZATION

A learning method generally categorized as a neural network approach is the self-organizing map conceived by Kohonen [4]. This method is based on the principle of competitive learning. A set of *codebook vectors* represent a multidimensional abstraction of the raw data that are supplied by the examples. The algorithm works on a multidimensional Euclidean space. The basic idea is to tune the structure of the codebooks conforming the signals that appear in discrete time steps. The closest codebook $m_i = m_i(t)$ to a signal (sample) $x_i = x_i(t)$ is updated following the rule:

$$m_c(t+1) = m_c(t) + \alpha(t) [x(t) - m_c(t)] \tag{2}$$

with $0 < \alpha(t) < 1$ being some gain coefficient, whereas the other codebooks that were not close to the sample remain the same:

$$m_i(t+1) = m_i(t)$$

In its supervised form the codebooks are learned by the so called Vector Quantization (VQ) [5]. In this case competitive learning is done with the known class membership of the samples. Several variants are proposed by Kohonen (LVQ1, LVQ2, LVQ3). A fixed number of initial codebooks is chosen. These codebooks are then updated using the competitive learning above. The codebooks are modified in the sense that they minimize an error criterion for the distance between the samples and their associated codebooks.

$Q^*$ shares some concepts with the VQ. From a set of raw samples a representative set of prototypes is generated. Prototype samples are dynamically updated, taking into consideration if the actual state of the system is capable to classify correctly. The difference however is that in VQ the number of the codebook vectors is fixed, where for $Q^*$ representative prototypes are dynamically created.

## 3.4 MULTIVARIATE ATTRIBUTE NODES IN DECISION TREES

Utgoff and Brodley showed with the PT2 algorithm [3] that it makes sense to test several attributes in a decision tree together. It is pointed out that eventually more compact trees are found compared to the conventional, univariate quantization method. The authors use linear threshold units (LTU) to separate classes. Symbolic attribute values are consequently half-spaces divided by hyperplanes. The trees for PT2 are always binary which implies a successive division of samples that could otherwise be divided directly if the separating boundary were not a hyperplane. Only two classes can be characterized.

The paper also points out the major drawback of multivariate splits mentioned before: decisions made by the system become less transparent. The trade-off between complex trees on one hand or complex tests on the other hand was described.

## 4. THE Q* QUANTIZATION ALGORITM

In this section we will specify in detail a technique that is conceived for the multivariate quantization of ordered numerical attributes. It should be understood as a possibility to implement the generation of symbolic attributes from a set of several continuous attributes. Other learning algorithms like Kohonen's LVQ or Salzberg's NGE could be used interchangeably for this purpose.

Q* has the following properties:

i.) It is non-parametric which is especially advantageous if the number of available training examples is small. For instance the number of prototypes for each class has not to be specified a priori.

ii.) It handles multimodal class distributions. Classes may be nested inside other classes in the multivariate continuous attribute space. This property is not true if e.g. a prototype is always calculated as the centroid of *all* samples that belong to a certain class.

iii.) It is self-organizing. No parameters have to specified by the user. Class concepts in the multidimensional continuous space are discovered automatically.

iv.) For the purpose of a preprocessor for symbolic Machine Learning algorithms, it reduces the dimension of the class description vector. Several continuous attributes are merged into one superattribute.

The subset of attributes that are numerical and continuous are represented by a set of prototypes (cluster centers) that are associated with the classes. The prototypes are created without any information about the other, symbolic attributes of the classes. The basic idea is to create a new prototype for a class whenever the actual set of prototypes is not capable of classifying the training set satisfactorily. This means that if a sample is mis-classified, we assume that it is not covered sufficiently by the prototypes of the true class of the sample. On the other hand if a sample is correctly classified by a prototype, it will influence the prototype itself. We use the method of updating a prototype by the *mean* of all samples which were correctly classified by the prototype. The mean is a multidimensional vector that is calculated by the individual means of the components of the vector. These components are the values of a certain continuous attribute. In the example of Fig. 1 for instance the mean of the three vectors for the *positive* class in the lower left corner of the graph

(10.1°C, 71.2%), (8.4°C, 75.2%) and (9.3°C, 78.2%) are represented by the mean vector (9.27°C, 74.87%) in Fig. 2.

This process of updating is repeated as long as the system commits errors in classification and as long as it dynamically changes the location of the prototypes.

The expectation for such a system is that it converges to a stable state in which the prototypes represent the classes. Multimodal class distributions should also be lear... d by the system without any prior specification by the user. The prototypes should adapt themselves to complex class distributions in the n-dimensional Euclidean space. Nested class clusters inside other class clusters must be detected. The system must also be deterministic, i.e. it must not cycle infinitely through its adaptation loop.

Table 3 gives the functional definition of the $Q^*$ quantization algorithm. Initially one arbitrary prototype represents one class. The whole sample set is presented to the system in one discrete time cycle. The feedback to the system is given by the classification of a sample. If the sample was correctly classified, join it to the list of positive examples of the nearest prototype. In the updating phase the mean value of this list will substitute the prototype. Hence the prototypes will gradually move to the centers of the local sample cluster that they should represent. Speaking in terms of probability distribution, the prototypes become the centroids of one mode of a multimodal class probability distribution.

On the other hand if a sample was "recognized" by a strange prototype (a prototype of another class than the correct class of the sample), then the algorithm concludes that the class of the sample is not sufficiently represented. Consequently the sample is declared to be a new prototype for the class that was not recognized correctly. This mechanism ensures that new prototypes are created as seeds for new clusters.

A highlight of the algorithm is that it needs no initial system parameter specifications like K-means or the similar ISODATA algorithm [13]. It adapts the prototype representation of the classes by trying to approximate the respective class regions in the multidimensional space.

One aspect of the clustering must be taken into consideration. The generation of the prototypes creates so called *outliers*, i.e. samples that are isolated from a cluster are always considered as a new cluster center. This should however be avoided in order not to overfit the clustering to noise, similar to the problem of the "pure" ID3 algorithm.

We therefore pass over the whole set of prototypes and delete those who never had any positive samples in their neighborhood. This procedure has the effect that outliers are purged and in overlapping areas of several classes the prototype occurrence is kept low.

## 5. EXPERIMENTAL RESULTS

The $Q^*$-algorithm will be used for experiments in inductive classification. First we will feed the quantization mechanism with the iris flower data set of Fisher [12] using a attribute vector reduced to 2 dimensions for the purpose of better visualization. This experiment serves for the illustration of the effects of $Q^*$. The second series of experiments will focus on the intentional conception of the algorithm, namely to be a preprocessor of numerical attributes for symbolic induction algorithms.

## Table 3: The Q* quantization algorithm

```
Let {S} be all labelled examples of the training set and {Sc} be the samples of
    class c.
Let {P} be the set of all prototypes and {Pc} be the set of all prototypes for
    class c.
Let {S+cj} be a list of positive examples that were correctly classified by Pcj.

Procedure Q*(S)
    Initialization. For each class c pick one aleatory sample Sc of that class as
    the first prototype Pc1 for that class c. Reset discrete time step: t ← 0.

    Repeat
        c ← first class;
        while ((c <= Nr. of classes) and (no mis-classification occurred))
            for (all samples Sc of class c)
                measure the Euclidean distance between Sc and all prototypes Pij of
                    all classes i, inclusively the class c;
                update the minimum distance (special case: do not update if minimum
                    distance is equal to measured distance and i is different from c);
                if (the closest prototype Pij to Sc belongs to class c, i.e. i=c)
                    join the sample Sc to the positive samples of Pcj:
                        {S+cj}←{S+cj}∪Sc;
                else
                    a mis-classification occurred;
                    the prototype Pkj with the minimum distance to Sc belongs to
                        another class k;
                    create a new prototype for class c: {Pc}←{Pc}∪Sc;
                    update all classes a by now visited: UPDATE_PROTOTYPES (a) with
                        a∈{1...c};
            increment c;
        if (no mis-classification occurred)
            for (all classes c)
                UPDATE_PROTOTYPES(c);
                test if prototypes change during the updating of class c
        next discrete time step t: t ← t+1;
    Until (the prototypes do not change anymore);

    Purge outliers. Delete all Pc which {S+cj} was always empty.

Procedure UPDATE_PROTOTYPES(c)
    for (all prototypes Pc of class c)
        if (list for positive examples {S+cj} of prototype Pcj is not empty)
            replace Pc by the mean of all S+cj;
            if (new value of Pc is different from old value of Pc)
                prototypes have changed;
            reset the list of positive examples for Pc: {S+cj}←{}
```

The ID3 decision tree generator program of Quinlan [1] will be the frame in which classification experiments will be carried out. Data from three real world domains is analyzed: i.) car imports ii.) hepatitis diagnosis and iii.) credit screening. The databases were downloaded from a repository for machine learning databases from the University of California at Irvine*. The databases were chosen in respect to the suitability for the quantization algorithm. The attributes are mixed, i.e. nominal and numerical. The experiments will compare the estimated classification error rate for the univariate (no multivariate quantization) to the error for multivariate quantization with Q*.

Beforehand we will define a set of conventions that were made at this stage of the system:

I.) The ID3 decision tree algorithm is used in its pure form. No pruning or soft thresholds

---

* directory: pub/machine-learning-databases    at server: ics.uci.edu

like described in [14] were applied to the decision tree. This constraint does not influence the comparison of univariate to multivariate quantization. ID3 is one of several possible frameworks for the symbolic induction.

II.) Examples with unknown values in one or more of their attributes were not used. The subset of the examples with only known attributes was used for training. The estimation of unknown attribute values lies outside the scope of this paper.

III.) Contradictions were discarded in the training set. If an identical attribute vector was representing two or more classes, it was deleted from the training set.

A few considerations about the time complexity of the algorithm: it is very hard to determine the analytic complexity of the $Q^*$-algorithm, because it depends on the topology of the data. Experimentally however even for large data sets it runs in a reasonable amount of time. The duration lies in the same dimensions as the ID3 tree induction. The example mentioned below of credit screening needs for 6 continuous attributes and 653 samples about 16 seconds on an IBM Risc/6000-32H workstation.

## 5.1 IRIS FLOWERS

The classical data set consists of 150 samples with 4 continuous valued attributed and 3 classes of flowers each being equally represented by 50 samples. We use this data set in order to permit a visual comparison of the learning algorithm with the method of the hyperrectangles proposed by Salzberg [11]. In his paper the author uses the 4th and 2nd attribute of the whole possible set of 4 continuous attributes. Figure 5 of his paper illustrate the effect of the hyperrectangle learning on the 3 classes. The "setosa" class is linearly separable from the other two classes using a linear decision boundary orthogonal to the "petal width" attribute. This fact is reflected by a single rectangle that defines the "setosas". For the other two classes "versicolor" and "virginica" the linear separation is impossible. The samples for the two classes overlap in attribute space. They must be approximated by several rectangles which also overlap partially.

$Q^*$ is submitted to the same data as in the hyperrectangle algorithm. Fig. 3 shows the result after the prototypes have been learned. For this illustration no outlier purging was performed. It is interesting to note that the easily separable "setosa" is represented by only 2 prototypes. The lower of the two prototypes would be deleted as an outlier if this part of the algorithm was executed. The hyperrectangle learning copes also easily with this class, because only one rectangle is needed to represent "setosa". The other two classes are harder to distinguish. Consequently more prototypes are generated were the two classes approximate. It can be observed that the prototypes align themselves to the regions which are formed by the raw data of the two classes "versicolor" and "virginica".

$Q^*$ learned 2, 18 and 11 prototypes, of these being 1, 1, and 3 outliers for the 3 classes "setosa", "versicolor" and "virginica" respectively. The induced ID3 tree consequently has 26 leaves with the multivariate attribute being the tested attribute in the root node.

## 5.2 CAR IMPORTS

This database is the number 4 of the repository of the University of California at Irvine mentioned above. It contains samples of imported cars and their specification. It was first used in [15]. The first attribute is the class attribute. It is called "symboling" and consists

of 6 different values which express a risk factor for assurance purposes. The $Q^*$ algorithm is used to join all continuous attributes to the new superattribute. From the total of 26 attributes these are nr.: 2,10-14,17,19-26 which are 15 continuous attributes.
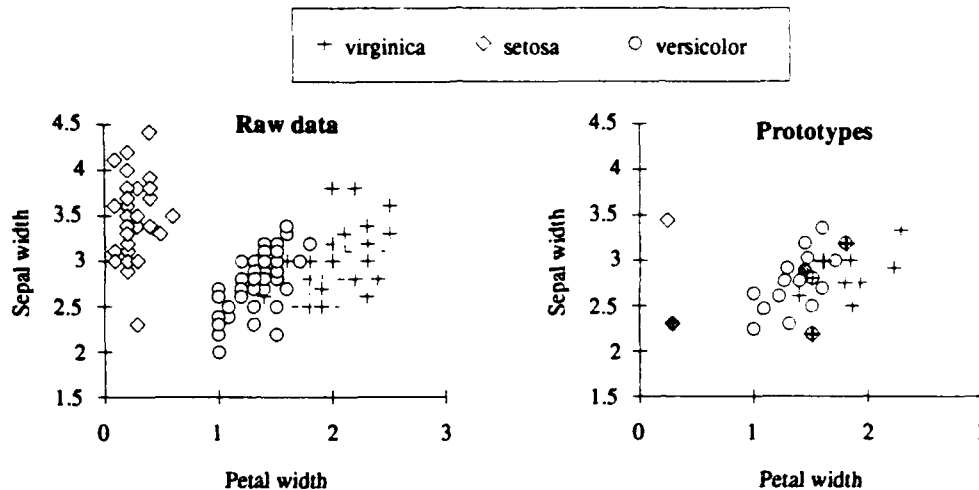


Fig. 3. The effect of the $Q^*$ quantization algorithm on the modified iris flower data: raw data and learned prototypes. The outliers of the respective classes are emphasized. The setosa class has 1 outlier, versicolor 1 outlier, virginica 3 outliers.

The error estimation is done by the *leave-one-out* method. From all the hold-out error estimation this is the most expensive but also the most unbiased. It can be shown experimentally that the leave-out-out can be applied in a reasonable amount of time. Consequently instead of splitting the data e.g. in 70% training samples and 30% test samples and calculating the average error for 5 runs, like in [8], we hold out one sample and build the classifier with the rest. This process is repeated for each sample and the average error is then the number of mis-classifications divided by the number of samples.

The univariate quantization was done for different interval lengths. Error rates were tested for uniformly split intervals of 0,1,2,3,4,5,10,15,20,25,30,40,45 and 50. The interval length 0 means simply that all continuous attributes for all samples get the same symbolic value; they are not relevant for classification anymore. From the original sample size of 205 those with unknown values were deleted which resulted in 159 samples.

The parameters that were compared are the principal attribute of the root node, its gain (ID3), the average number of leaves of the tree and the estimated error. Fig. 4 shows the multivariate error rates of $Q^*$ compared to the univariate error rates for different interval coarsenesses. It can be observed that the multivariate error is 22.0% compared to the lowest univariate error of 13.84% for an interval coarseness of 20 discrete steps for all originally continuous attributes and the highest error rate for binary intervals of 24.53%. In relation to the complexity of the decision trees the following can be stated: For the 160 error estimation runs, the average complexity of the multivariate ID3 tree is slightly higher than for the best univariate tree. The number of leaves for the multivariate tree lies between 64 and 70, where on the other hand the number of leaves for the univariate case is between 54 and 58 for the coarseness with the lowest error rate. Multivariate: the root

node of the ID3 is 17 times the superattribute and 143 times the second (nominal) attribute. The gain in the root node varies between 1.92 and 2.18. Univariate: the attribute with the highest gain in the root node is always the attribute 10 with a gain between 1.18 and 1.22. For this database the multivariate quantization has a lower error rate only for one case of the univariate quantization. The high gain of the superattribute can be explained by the fact that almost each class has an exclusive set of values for the superattribute.
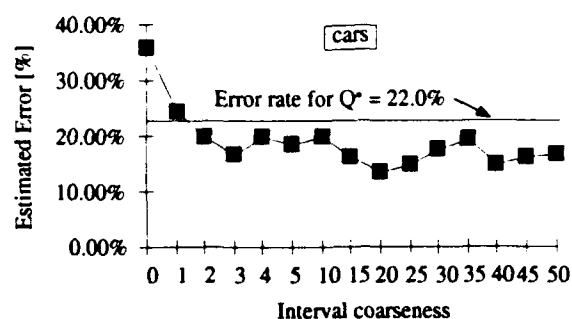


Fig. 4. Estimated error by the leave-one-out method of the ID3 classification tree for the car import database with univariate quantization of continuous attributes and multivariate quantization using Q*.

## 5.3 HEPATITIS DATABASE

This data set was used in [16] and [17]. After deleting the samples with one or more unknown attribute values, 80 of the originally 155 samples remained. Attributes 2 and 15-19 were merged into the superattribute.
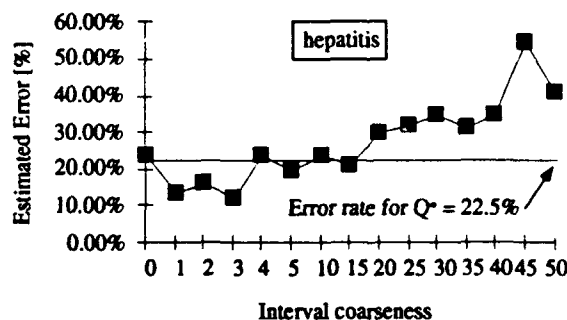


Fig. 5. Estimated error for the hepatitis database. Analogous case like in Fig. 4.

The error for the multivariate quantization is 22.5%, whereas the lowest univariate error is 12.5% for the size 4 of the discretization interval. For 10 out of the 15 different univariate coarseness intervals Q* classifies better, see Fig. 5.

The gain for the multivariate case falls between 0.35 and 0.51, with the superattribute always being the winner at the root node. The number of leaves varies between 29 and 37.

In the best univariate case attribute nr. 20 (histology) wins once with a gain of 0.22, attribute 19 (protime) wins two times in the root node of ID3 with a gain of 0.17 and 0.19. The remaining 77 times attribute 18 (albumin) wins with a gain between 0.16 and 0.24.

## 5.4 CREDIT SCREENING

It depends on several conditions whether a person is granted a credit card or not. In this database provided by Quinlan [18], [19] a history of credit card allowance decisions was recorded. The original data set of 690 was reduced to 653, deleting unknown samples. Continuous attributes are 2, 3, 8, 11, 14 and 15. The error estimation was not done by the leave-one-out. Instead the average of 5 test runs with a split of 70 to 30 for training and test was taken to estimate the error rate.

Observing the graph in Fig. 6 we can conclude a very interesting property of this data set. The lowest univariate error appears for the discrete interval step 0. This means without the use of any numerical attribute it is possible to decide best if a person can be granted a credit card or not. Any additional numerical attribute is more confusing than helpful in the decision making process. This emphasizes the necessity of an essential preprocessing step in classification: *feature (attribute) selection*.

As soon as a numerical attribute is included, the error rate starts to raise. Logically one cannot expect from the $Q^*$ algorithm that it classifies better than an univariate technique for this database.

The average multivariate error is 32.4% with a gain of 0.83 to 0.94 always for the superattribute as the root node winner and the number of leaves between 317 and 330.

In all cases of the univariate version the attribute number 9 (binary with values t, f) is the winner of the root with gain values between 0.4 and 0.52. The number of leaves is with values between 94 and 257 always smaller than for the multivariate case.
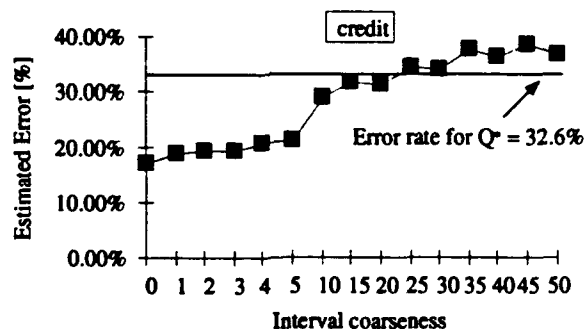


Fig. 6. Estimated error for the credit card granting database by splitting the data set into 70% training samples and 30% test samples using the average of 5 test runs.

## 6. DISCUSSION AND CONCLUSIONS

We have presented a method for the multivariate quantization of ordered numerical attributes. The quantized values serve as symbolic values for Machine Learning induction algorithms. The $Q^*$ algorithm clusters the samples of a data set in the subspace of the numerical attributes. It prevents the user from dividing the ordered attribute scales into discrete steps. Its advantage is the parameter free nature of the approach. It is self organizing and can be used as a straightforward preprocessor for Machine Learning algorithms.

The drawback of such a multivariate method is that decision structures become less transparent.

The results show higher recognition rates in some cases but do not yet satisfy. One

would expect a greater impact on the accuracy of the classifier. Perhaps other symbolic induction algorithms than ID3 are fitter for the method. Future research will try to focus on that question.

It also seems that the gain for the superattribute is always to high in the root node, so that other attributes never have a chance to win in the root node. By further reduction of the number of prototypes for the samples it can eventually be achieved that the gains for the superattribute and the other symbolic attributes approximate in order to permit a fairer competition. The gain also favors attributes with many different values. These facts are open questions which have to be investigated in future work.

## REFERENCES

[1]   J. R. Quinlan,   "Discovering rules by induction from large collections of examples," in D. Michie (Ed.),   *Expert systems in the micro electronic age*, Edinburgh University Press, 1979.

[2]   ——,   "Induction of decision trees," in *Machine Learning* 1:81-106, Kluwer Acad. Publ., Boston, 1986.

[3]   P. E. Utgoff and C. E. Brodley,   "An incremental method for finding multivariate splits for decision trees," in *Proc. of 7th Int. Conf. on Machine Learning*, Austin, Texas, USA. Porter, B. and Mooney R. (eds.), 1990.

[4]   T. Kohonen,   "The self-organizing map," in *Proc. of the IEEE*, Vol. 78, no. 9, Sept. 1990.

[5]   ——,   "Learning Vector Quantization," Helsinki University of Technology, Lab. of Comp. and Information Science, Report TKK-F-A-601, 1986.

[6]   I. Kononenko, I. Bratko and E. Roskar,   "Experiments in automatic learning of medical diagnostic rules," *Technical Report*, Ljubljana, Slovenia: E. Kardelj University, Faculty of Electrical Engineering, 1984.

[7]   R. S. Michalski and J. B. Larson,   "Selection of most representative training examples and incremental hypotheses: the underlying methodology and the description of programs AESEL and AQ11V," *Research report UIUCDCS-R 78-867*, University of Urbana-Champaign, Illinois, USA, 1978.

[8]   P. Clark and T. Niblett,   "The CN2 induction algorithm," in *Machine Learning* 3:261-283, Kluwer Acad. Publ., Boston, 1989.

[9]   J. Catlett,   "On changing continuous attributes into ordered discrete attributes," in *Proc. of EWSL-91: European workshop on Learning*, Porto, Portugal. Kodratoff, Y. (ed.), 1991.

[10]  L. Rendell,   "A general framework for induction and a study of selective induction," in *Machine Learning* 1:177-226, Kluwer Acad. Publ., Boston, 1986.

[11]  S. Salzberg,   "A nearest hyperrectangle learning method" in *Machine Learning* 6:251-276, Kluwer Acad. Publ., Boston, 1991.

[12]  R. Fisher,   "The use of multiple measurements in taxonomic problems," in Annals of Eugenics 7, 179-188, 1936.

[13]  J. T. Tou and R. C. Gonzalez,   "Pattern recognition principles," Reading: Addison-Wesley, 1974.

[14]  J. R. Quinlan,   "Decision trees as probabilistic classifiers," in *Proc. of 4th Int. Workshop on Machine Learning*, Irvine, California, USA. Langley, P. (ed.), 1987.

[15]  D. Kibler, D. W. Aha and M. Albert,   "Instance-based prediction of real-valued attributes," in *Computational Intelligence*, 51-57, 1989.

[16]  P. Diaconis and B. Efron,   "Computer-intensive methods in statistics," in *Scientific American*, Vol. 248, 1983.

[17]  G. Cestnik, I. Konenenko and I. Bratko,   "Assistant-86: A knowledge-elicitation tool for sophisticated users," in I. Bratko and N. Lavrac (Eds.) *Progress in Machine Learning*, 31-45, Sigma Press, 1987.

[18]  J. R. Quinlan,   "Simplifying decision trees," in *Int. Journal of Man-Machine Studies* 27, pp. 221-234, Dec. 1987.

[19]  ——,   "C 4.5: Programs for Machine Learning", Morgan Kaufmann, Oct 1992.

# UTILIZATION OF THE CASE-BASED REASONING METHOD TO RESOLVE DYNAMIC PROBLEMS

Sophie Rougegrez
Laforia-IBP
Université Pierre & Marie Curie
4, place Jussieu
75230 Paris cedex 05
France
e-mail : rougegre@laforia.ibp.fr

## ABSTRACT

Case-based reasoning allows to resolve problems by comparison with already resolved ones. This technique is being utilized for domains in which little knowledge is correctly formalized.
But a minimum of knowledge has to be available to represent cases stored in memory by pertinent features. But this one does not always exist, as we shall see.
The method we use makes smaller this necessity. In this paper we describe the system conceived. Thanks to the definition of viewpoints, this one permits to predict an evolution from a given date.

## INTRODUCTION

Case-based reasoning is a technique of resolution used when the knowledge required for resolution is insufficient or imprecise. It consists in using similar problem(s) already resolved. It avoids rebuilding a solution when a similar problem has already been handled. In this kind of reasoning, the entity manipulated is named "case", associating the statement of the problem with its solution. Cases are stored in a memory, called case base [1].

The process of case-based reasoning breaks down into the following steps :
1- definition of the problem by its representation in a form adapted to the reasoning,
2- search of the pertinent cases in the case memory for the resolution of that problem. This phasis is generally carried out in two steps : the first consists in selecting in memory a set of cases susceptible of being interesting , then among those to choose the best(s) for the resolution to carry out. They are the source cases.
3- building of the solution :
The question is to transfer the solution from the chosen case to the case to fill up, the target case. But very often, there's no identity relation between both and the transfer is then followed by an adaptation step.
4- validation :

Once the solution is worked out, the question is to test it. This step is performed when the resolution is turned onto a precise objective, a goal satisfaction for example. When the reasoning has failed, it can be followed by the correction of the knowledge being used [2].

The CBR *sine qua non* condition is then the possibility of defining a case entity made up of the terms problem and its solution. It has been shown that this kind of reasoning could be used in many areas : cookery [2], [3], law [4], planification [5], [6], [2], etc.
We have used case-based reasoning for the determination of evolutions from a given date, for areas in which very little knowledge is available. Our approach is new for two reasons :
        - our case-based reasoning manipulates temporal data,
        - our case-based reasoning doesn't require the use of important features to identify the cases.

We will describe in section I more precisely the problem proposed. In section II we shall detail the reasoning steps. We will proceed in the third part by an area description which has been the subject-matter of an application, and in section IV by the presentation of the system conceived, illustrated by the description of one of its components and we'll conclude.


## 1. THE PROBLEM PROPOSED

We wish to be able to determine the follow-up of an evolution. An evolution can be described this way : an event which triggers it and other events that follow it. An event can result further to one or several others. But an event can happen regardless of the others too. Our definition of an event is the following : a part change of situation at a given instant, without information about its life time.

We have especially interested ourselves in evolutions for which we have no knowledge about the links between the diverse events which make it up. The use of "classic" reasoning methods as well as expert systems was then not considerable. Therefore we have chosen the case-based reasoning method.

From an evolution described by a set of events which occured between an instant $t_0$ and an instant t, we wish to know which events are going to take place afterwards. Hence, we search for a similar evolution, that is to say an evolution which has experienced the same events at a given time. Then we utilize those which followed it to work out the searched evolution.

The fact of ignoring the causal links between events doesn't permit us to "abstract" from the set of events describing an evolution a set of features, as it is

traditionaly done in case-based reasoning. We are thus compelled to match cases per the consideration of the totality of events that have occured.


## 2. THE REASONING STEPS

The "important feature" concept is fundamental in case-based reasoning. That's those features which permit to represent, store, and retrieve cases.
Case-based reasoning is quite often opposed to rule-based reasoning. We blame this one for being not the perfect reflect of the expert knowledge. The use of problem resolution experiences permits indeed to palliate this drawback [7]... on the following single condition : the features identifying them are the reflect of the important elements effectively characterizing the experiences.

There exists indeed domains in which the experiences only can be used. That's the case with the areas in which we have interested. The expertise is so little known that we don't have even the means of associating those experiences with a set of important features .
The broad lines of our case-based reasoning steps are consequently the following.


### 2.1. CASE SELECTION

Several systems select all cases in memory, this is the case with PROXIMITY, GROWTH and SHRINK [8].

But most of the time, a few cases only are selected : the selection of potentialy interesting cases is then followed by the choice of the most interesting cases, by a constraint satisfaction algorithm for example [9] or the use of a model [10]. In HYPO [4], the method being utilized is quite different : instead of reducing the set of selected cases, the system widens it. It doesn't attempt to select the most interesting cases but to find all cases having a link with the target case.

Our approach consists in reducing the set of the potentialy candidate cases to a singleton : the one in which we could locate a given follow-up of events.


### 2.2. MATCHING OF CASES

Given that we do not have at our disposal the sufficient knowledge to abstract the important events from an evolution, which would facilitate the comparison of evolutions, we consider the totality of those events.

Moreover, there may have different kinds of events. For example, let's consider the case of a person who is ill. The evolution of her state results from the value of

several factors, specially their respective evolutions. To settle a diagnostic, it may be interesting bringing together the evolution of the state of the ill person and the evolution of well-known illnesses. Now, the comparison of two illnesses evolutions is hardly conceivable by considering them in their whole : how can we consider the evolution of the blood pressure of the patient with the one of its sedimentation speed ?

This remark applies equally to natural catastrophes, chosen area, and more generally to every phenomenon in which a global state results of a large number of parameters. These thoughts have led us to consider the evolution according to a viewpoint, this one being associated with a given type of events.

Two evolutions are thus similar iff :
*For a given follow-up of events relative to the evolution E and the viewpoint P, which occured between instants $t_0$ and $t_1$, there exists two instants $t'_1$ and $t'_2$, between which events that occured according to the viewpoint P in the evolution E', are similar to E between $t_0$ and t.*

## 2.3. EVOLUTION DETERMINATION AND ADAPTATION

The evolution produced by our system consists in events coming from the source case, that is to say the chosen evolution. The adaptation step consists in adapting the events to the target case. That will be described more precisely in the follow-up.

To realize these steps, all the events have been classified according to their nature and have been associated with different viewpoints. But we have to choose which kind(s) of events(s) is (resp. are) going to be used to translate the hypotheses about the evolution. Hence, we have proceeded to an other categorization of events according to the kind of links which associate them to others. Hence, we distinguish :
- events which result of the influence of other events,
- events which influence other events.

That is to say we suppose that we know among all the events which occur, those who indicate the influence of some other events, and those who influence others. That's the minimal knowledge that we use in our reasoning. We represent then our hypotheses in terms of events which are subject to the influence of others. In medical diagnosis, such an hypothesis may be "the patient will have a crisis at ten", this one resulting of events like a drop of the blood pressure.

We have considered this approach in the frame of a real time system. At any moment of the evolution, it contains the following information about the evolution :
- the description of the evolution which is occuring,

- the hypotheses realized about its future, each hypothesis being linked to a viewpoint, that is to say a kind of parameter.

We have defined a multi-expert distributed architecture, composed of modules associated to a kind of events, these events having an influence on others ; and a database containing on the one hand the information received about the evolution considered, and on the other hand the hypotheses resulting from the reasonings done by each module. These latest are thus translated in terms of "events which are influenced by" other events. Each module reasons per case to build hypotheses relative to the future evolution, according to the viewpoint which interests it.

That's the area of forest fires which has made up the point of departure of our work . We describe below the area and how we have implemented the proposed method.

## 3. THE DOMAIN

The propagation of a forest fire is a complicated phenomenon. It depends on a high number of environmental factors : wind, relief, vegetation, dryness, etc.

A forest fire is described by overall information such as : "the risk of fire start in the day of the 15 july was very strong". Information can be much more precise as "an attack group arrived on the Ste Victoire mountain at 4:12 pm ". Precise information correspond to what we designed above as events.

The set of parameters, like relief and vegetation run along, the arrival of engins constitute also events. In our object representation of a case, the relief and the vegetation are yet information independent of time. But this is not the case with their interpretation. Each relief object, describing a point located at a given altitude, is indeed represented in the case because at a given moment, fire went across it. Our representation permits doing the link between these "timeless" pieces of information and time (see appendix). Therefore we will expand our definition of an event to every change linked directly or indirectly to this one.

These parameters are going to have an influence on the fire propagation and then on the description of the fire evolution, in terms of events describing its spread. But the link between parameters and the propagation is far away from being explicit.

Therefore, we have chosen to compare forest fires viewpoint per viewpoint. In fact, in order to determine the fire propagation which is occuring, we are going to search for similar fires according to the three viewpoints

described above : wind, relief and vegetation. More precisely, under the wind viewpoint, we are going to find the fire which has known the same wind changes : direction and speed. For the relief, we are going to search for a fire which has spread over a relief identical to the fire which is occuring, it is the same for the vegetation. And the established evolution will be translated in terms of events describing the propagation.

## 4. THE SYSTEM

Its objective is then being able to predict the evolution of a running fire. This system has a distributed architecture as figure 1 below shows it. Each of the module reasons per case according to a fire parameter : relief, vegetation, or wind. The hypotheses, generated by each of the modules are added to the database. Hypotheses are represented by events describing the propagation. That's indeed this kind of event that interests us.

The system is a real-time system which reacts to new information that it receives, among others, on the fire progression. Afterwards, theses ones allow generating hypotheses on its future propagation. The relief module for example, is going to search for a fire having run along the same relief in the memory of cases, and is going to generate hypotheses on the future fire propagation. It is likewise for the other parameters.



Fig. 1 : System architecture
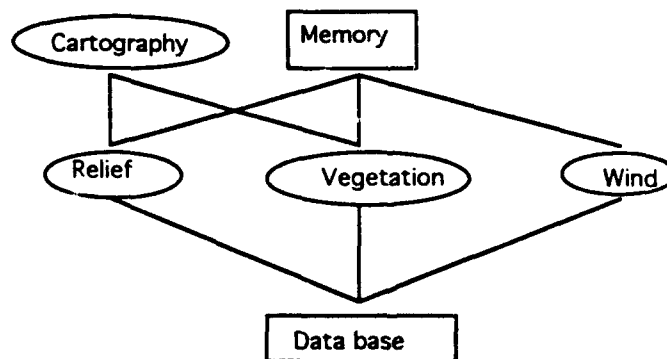
The database contains data that can be :
- deduced by the modules associated with the diverse parameters. They are the hypotheses on the propagation : "fire will be at the farm Cazeneuve at 4:30 pm",
- entered by the user : changes of wind which happened, information received on the actual spread of fire,
- deduced from the data provided by the user : relief and vegetation run along.

We consider indeed that these latest data can be deduced from the information on the fire advance, by the examination of a ground map. That's the role of the module "cartography".

We describe now the working of the relief module, which makes up itself a case-based reasoning system.

*Relief module :*

From a piece of information concerning the fire propagation, this module searches for a fire in memory which at a given moment, has run along the same relief as the one run along by the target fire since its outbreak. But that's not enough.

The relief that the fire runs along can indeed have long term effects on its propagation. For example, it may have accelerated it. But its effects can be immediate : a propagation stop caused by the descent of a strong slope, for example.

If we wish to generate hypotheses about the future fire behavior, we don't have to consider all the relief that it ran along only, but also the relief that it's going to run along. Therefore, we have to anticipate the relief that will be run along.

Figure 2 presents the information utilized to calculate the propagation of the target fire. $E_f$ and $E_s$ are events describing the fire propagation. The "target fire" is the fire which is occuring. We have described above the profile of the relief that it ran along since its outbreak, that is to say until the t instant

We have represented the "anticipated relief" too, represented after the t instant here. We have associated to the time axis events corresponding to the piece of information that we have on the fire propagation. $E_f$ is the latest event. In our representation, we separate the different kinds of information. As it is shown in the diagram, the time axis permits yet to link them (see appendix).

The principle of our method is to retrieve a fire in memory which, under the relief viewpoint, is similar to the target fire. We search then for a fire which has at a given moment run along the same relief : the source fire above has well went across a similar relief, between the instants $t'_1$ and $t'_2$. The event $E_s$ has occured when the fire has gone across the relief we have anticipated : it results therefore of all the relief run along and at that moment. These are the events we are going to utilize to describe the hypotheses about the target fire propagation.
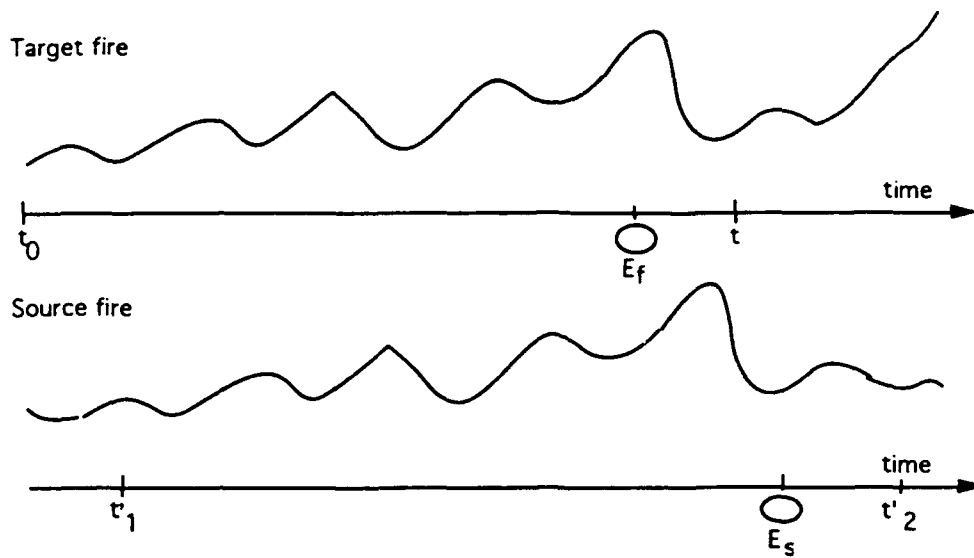
Fig. 2 : Evolution prediction according to the relief viewpoint

We descrive below the reasoning steps :
- transformation of the relief under a suited shape,
- retrieval of a fire having run along the same relief,
- retrieval of information about its propagation,
- transfer and adaptation of this piece of information towards the target fire,
- adding of the hypotheses carried out to the database.


## 4.1. RELIEF TRANSFORMATION

The relief called up in the fires is represented under the form of a curve. In our system, by contrast with the description of fires as it is done by the experts, it consists into a succession of points faraway from each other of a certain distance and located at a given altitude. The relief module is going to try to retrieve a fire which has spread over the same relief. The mathematical methods grounded on curves differences are not useful here : theses ones consider a pixel follow-up. Hence, we try to compare slope successions. The likeness criterion of two profiles is indeed the sense of the slope (ascendant, descendant), the order in which they have been went across, and too the eventual passing over of more complicated relief accidents too : such as a pass, a valley, etc.

From the numerical relief representation under a numerical form, we build a symbolic representation. This is equivalent to slipping the curve into a succession of segments, to associate those with a slope and if possible, replacing a slope follow-up by a "shape" : for example, a valley.

We dispose then of two representation levels : slopes and shapes. The transformed relief will be represented thanks to both levels.

In the expert language, we speak about "weak slope", "strong slope", etc. This vocabulary has led us to classify the set of slopes : to each group of slopes, we associate a representative, a prototype in form recognition [12]. We dispose then of slope models, either ascendant, or descendant. The degree associated with each is symbolic. Indeed we have used the multi-set theory [11] according to which an element belongs to a set to a certain degree.

A first transformation from a set of points consists then in generating slopes in the form of symbols with which we associate a degree. From these symbolic slopes, we try to generate forms, associated with a degree too. This latest is a vector constituted of degrees associated with slopes constituting the generated form.

## 4.2. RETRIEVAL OF A FIRE HAVING RUN ALONG THE SAME RELIEF

Traditionaly in CBR, the selection of the most comparable case to the source target is carried out by the determination of a set of "candidate" cases, then among those the selection of the best. We consider that all cases are candidates in the beginning. We evaluate the likeness with the target case and only then we choose the best one.

Our matching algorithm is hence put in charge of matching lists of symbols representing slopes or forms, associated with a membership degree which can be a symbol, or a vector of symbols.
For example : (ascendant slope, strong)-(descendant slope, weak)-(descendant slope, very weak), (valley, (weak, strong)).

The matching step compares then two symbol follow-up associated with a membership degree. Two follow-up look like each other if :
- they are constituted of the same elements,
- the elements follow each other in the same order.

These two matching criterions constitute two different methods, two viewpoints, which are going to me mutually utilized to evaluate this one. From a matching results then a couple of costs : the first value comes from the comparison between successions as sets of symbols. If the sets are composed with the same elements, the matching cost is null. The second considers the positioning of the elements in both successions respectively. Figure 3 describes the corresponding algorithm.

```
ltarget _ symbol list describing the relief stemming from the target case.
lsource _ symbol list describing the relief stemming from the source
case.
cpt _ 0.
while ltarget and lsource are not empty do
      ftarget _ first (ltarget). ltarget _ rest (ltarget).
      fsource _ firsts (lsource). lsource _ rest (lsource).
      if ftarget and fsource describe the same kind of slope (ascendant
      or descendant) or the same kind of form then
          cost _ 0
      else
          search for a kind of slope or the kind of form the most like fsource
          and such that it is in ltarget.
         The found object is aux. We take it off from ltarget.
         cost _ the distance, in number of positions, from the aux position
          in ltarget to the ftarget position in ltarget + the "virtual distance" which
          exists between fsource and aux.
      end if
end while
result _ cpt.
```

**Fig. 3. Matching algorithm between two relief descriptions**

The determination of the most similar slope or shape is realized from a graph whose values are symbols. Each value is related to the "most proximate" symbols per bow stamped with a "virtual distance" which separate them. We have defined the proximity between two symbols by the similitude of their effects on the fire.

We dispose at this stage of a list of cost couples. As theses measures are done for each case in memory, there are as many as there are cases in memory. The best case is the one whose two costs are lower than those of all the others. In the absence of a case presenting this feature, we'll take the one whose one of costs is lower than all the others, independently of the couple it belongs to.

## 4.3. SEARCH OF INFORMATION ABOUT THE PROPAGATION

The best case being chosen, we can proceed to the calculus of the fire evolution. This one is realized from information about the propagation of the source fire which has been selected. Events in a case being classified in chronological order, we extract from a case the following events : the event $E_S$ in figure 2.

## 4.4. TRANSFER AND ADAPTATION OF EVENTS TO THE TARGET CASE

Once these events have been selected, we have to adapt them to the target case. We consider that each event is represented in a plan guided by a time axis and a distance

axis, whose origin corresponds to the location and starting date of the fire. The transfer and adaptation of an event consist then in a transposition of a guiding into another one.

## 4.5. ADDING OF THE GENERATED HYPOTHESES TO THE DATABASE

The hypothetical events carried out describe the succession of the running evolution. All the data on the database are organized along the structure of a case defined in appendix. At every moment of the reasoning, the data base is then quite readable. We only have to suppress all the hypothetical data to store the new case in memory.

There is another step in case-based reasoning systems that doesn't exist in all systems. It is the evaluation phasis. This one depends indeed on the available knowledge to validate the yeld results. When the reasoning is directed towards a single, or several goals, it suffices to verify that those are reached [2]. In all other cases, if it was possible to evaluate the result, the reasoning would have no sense anymore, at least in the frame of a reasoning use to palliate the lack of expertise. In our case, experts themselves can't evaluate the results yeld. Only a comparison with the actual evolution has some sense.

## CONCLUSION

The determination of the evolution of a running process is a new problem, at least in case-based reasoning. The problem of the determination of a follow-up has been handled, but for well-defined data between which we can find a relation [13].

The determination of evolutions, much more general, has constituted the target of our research. Our aim was to conceive a system being able to determine its "future". An evolution is described thanks to a set of events about which we have little information : we don't know how they interact and then how they influence the evolution. We only know that they are of different kinds and that, inside each group, events either have an action on the events of other event categories, that is to say that the second ones result from the first ones, or by contrast undergo the other events.

We have introduced the notion of viewpoint on an evolution : we do hypotheses on its future along an event category. Hence, we have defined a distributed architecture in which each module reasons per case according to a viewpoint.

Our approach is original because each of our cases constitutes a direct and integral transcription of the evolution representations on which we have based our

system. Our matching method furthermore is very little domain dependant.

Our approach has yet an inconvenient which is the necessity of considering all the cases. But the use of an index requires a case interpretation in terms of features. Given the lack of knowledge about the considered area, this one would have all the chances to be erroneous. That's what we firmly wanted to avoid.

We study now the feasability of the combination of hypotheses generated by each of the modules. That's a difficult problem because the events we use to build evolution hypotheses result "intrinsically" of the influence of all the kinds of events. Moreover we obtained them by considering only one of them. In short, we should retire from hypotheses carried out all the influence of events which don't have been voluntarily considered. An other solution, much more reasonable, would consist in controling the choice of cases by each module : if we succeed in finding a case which is the best along all the viewpoints, the problem is actually resolved.

Appendix

## CASE REPRESENTATION

A case is described by much information : general comments, such as for example, the eclosion risk the day of the catastrophe and information relative to the propagation of the fire too. That's these ones that interest us. We have represented them in figure 4.
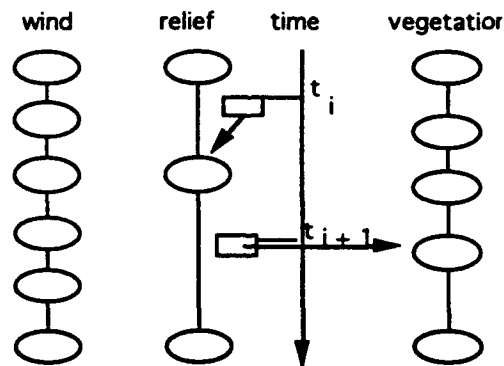


Fig. 4. Structure of a case

They are classified according to their nature and constitute thus the different viewpoints described before : wind, relief, vegetation. Inside each of them, some events have happened. Each event is an object related to the object which follows it in time and the one which precedes it. But the object contains no temporal information. Inside each viewpoint, we have an event chaining.

To be able to do the link between the different viewpoints, as it is necessary for the construction of hypotheses, our ca·es contain a temporal axis materialized by an instant chaining, each instant being linked to all events which happened at this date.

However, all the events can not be associated with an instant. For the relief for example, an event describes a passed over point : the distance which separates it from the fire departure, its altitude. Il is hardly considerable to know at each moment two points, separated from each other of a few meters have been reached !

We palliate to that information incompleteness b, the description of a fire the following way :

108

for each event E, whatever the considered viewpoint, our case representation ensures us that there are at least two events E1 and E2 such as :
     - $E_1$ is a predecessor of E and $E_2$ is a successor of E,
     - $E_1$ and $E_2$ are associated to an instant on the time axis.

This manner, an information, described independantly of time, can be associated with it.

BIBLIOGRAPHY

[1] : SLADE Stephen (1991):
*Case-based reasoning : a research paradigm*
AI magazine, spring 1991.

[2] : HAMMOND Kristian J. (1990) :
     *Case-based planning : a framework for planning from experience*
Cognitive science n°14, pp 385-443.

[3] : Janet L. KOLODNER (1987) :
     *Extending problem solver capabilities through case-based inference*
Proceedings of the 4th International Workshop on machine learning, pp 167-178

[4] : Edwina L. RISSLAND et Kevin D. ASHLEY (1986) :
*Hypotheticals as heuristic device*
Proceedings of AAAI, pp 289-297.

[5] : ALTERMAN R. (1986) :
     *An adaptative planner*
Proceedings of AAAI, pp 65-69.

[6] : KLEIN Gary A., WHITAKER Leslie A., KING , JAMES A. (1988)
     *using analogues to predict and plan*
Proceedings of the second Workshop on CBR, Pencola Beach, FL.

[7] : C. SCHANK & C.K. RIESBECK (1989) :
*Inside Case-based reasoning*
Lawrence Erlbaum Associates.

[8] : D. KIBLER, D.W. AHA (1987) :
     *Learning representative exemplars of concepts. An initial case study*
Proceedings of the 4th International Workshop on machine learning, pp 24-30.

[9] : Janet L. KOLODNER (1989) :
  *Judging which is the "best" case for a case-based reasoner*
Proceedings of the second Workshop on Case-based Reasoning,
Pencola Beach, FL.

[10] : BICHINDARITZ Isabelle, SEROUSSI Brigitte (1991) :
  *Validité a priori des inférences produites par un raisonnement à partir de cas : objectivation de l'analogie par des critères de ressemblance fonctionnellement quantifiés,*
Proceedings of RFIA.

[11] : H. AKDAG, M. DE GLAS, D. PACHOLCZYCK (1990) :
*Towards a qualitative theory of uncertainty*
technical report of LAFORIA-Université Paris 6, 8/90, 1990.

[12] : Jean-Claude SIMON (1984)
  *La reconnaissance des formes par algorithme*
Masson.

[13] : Thomas G. DIETTERICH, Ryszard S. MICHALSKI (1989) :
  *learning to predict sequences*
in Machine learning, vol II, Morgan Kaufman.

# FORMALIZATION OF AN ONTOLOGY OF CERAMIC SCIENCE IN CLASSIC

**Piet-Hein Speel, Paul E. van der Vet, Wilco ter Stal & Nicolaas J.I. Mars**

**Knowledge-Based Systems Group, University of Twente**
**P.O. Box 217, 7500 AE Enschede, The Netherlands**
**email: {speel, vet, terstal, mars}@cs.utwente.nl**

## ABSTRACT

Recently, the term "ontology" has become popular in the field of AI. However, often confusion arises when one is attempting to relate an ontology to existing AI subjects. We argue that an ontology (i) is situated within the conceptualization stage of knowledge-based system (KBS) development methods, (ii) specifies the domain on which logic-based knowledge representation (KR) languages are defined semantically, and (iii) represents the terminological/description (and not the assertional) component.

We have actually developed an ontology for a subdomain of materials science, called ceramic science. We have formalized and implemented it in the CLASSIC KR language to allow use of this ontology within the Plinius project. (The aim of the Plinius project, currently undertaken by our group, is to build a system which is able to semi-automatically extract knowledge from natural-language texts concerning ceramics.)

In this paper, we first concentrate on the integration of the three previously mentioned AI subjects and the face of an ontology in this integrated whole. Then, we focus on both the syntactical and semantical formalization of the Plinius ontology in the CLASSIC KR language. We argue that the semantics of the formulae need to be defined in order to guarantee the correctness of these formulae with respect to a model which includes the Plinius ontology.

## 1. INTRODUCTION

In the last two decades, many expert systems and knowledge-based systems (KBSs) have been built. However, these KBSs are task dependent and rather small. In order to build KBSs which cover a realistic complex domain (which results in a large KB) or which contain more than one KB, an ontology might be used. Examples of projects in which an ontology is used are LILOG [1], UMLS [2] and Cyc [3].

Briefly, an *ontology* consists of a conceptual vocabulary of a particular domain together with a general framework in which the dependencies between the concepts of the vocabulary are explicated. In this paper, we concentrate on a method to formalize an ontology. A formalized ontology is called *ontology KB* in the remainder of this paper. We do not concentrate on the design of an ontology which has been discussed in [4–6].

This paper is organized as follows. In §2, we give an overview of AI subjects which are relevant for the formalization of an ontology. We integrate the relevant parts of these subjects and place the ontology and ontology KB within the integrated whole. Then, we apply the resulting method to the Plinius project which is briefly described in §3. In §4, we discuss a part of the Plinius ontology and in §5, we describe a part of the CLASSIC KR language. In this language, we have formalized the Plinius ontology, which we present in §6. The implementation of the Plinius ontology KB and the application of the Plinius ontology to the other Plinius domain specific KBs is given in [7].

## 2. RELEVANT AI SUBJECTS FOR THE DEVELOPMENT OF AN ONTOLOGY KB

In this section, our objective is to give a brief overview of the AI subjects which are relevant for building an ontology KB for a particular domain. In §2.1, the stages within KBS development methods are introduced and the relevant stages are selected. In §2.2 and §2.3, we discuss the syntax and semantics of logic-based knowledge representation (KR) languages. In addition, we relate the syntax and semantics to two relevant stages introduced in §2.1. In §2.4, we describe the subdivision of domain knowledge into a terminological and an assertional component. Then, in §2.5, we describe the term "ontology" and we relate it to the relevant issues of the previously discussed AI subjects.

### 2.1 KBS DEVELOPMENT FOR BUILDING AN ONTOLOGY KB

Within AI, research has been done in developing methods for building KBSs or expert systems. Usually, such development methods contain a fixed sequence of stages [8]. The relevant stages for building an ontology KB for a particular domain are elicitation, conceptualization, formalization, implementation, and testing. At the *elicitation* stage, the knowledge of the domain is elicitated in an informal, usually verbal, form. At the *conceptualization* or *analysis* stage, the elicitated data is interpreted, which means that the concepts are analysed and their implicit relationships are made explicit. This results in the construction of a precise description of a *conceptual base*. The *formalization* stage involves the translation of the conceptual base into a formal representation in a KR language. Examples of such languages are order-sorted logics and descendents of the KL-ONE family of KR languages [9]. The structure of this formal representation should be manageable and maintainable. In addition, it should be possible to use the inference mechanism of the KR language effectively and efficiently. At the *implementation* stage, the formal representation is implemented in a language which is executable by a computer. Finally, at the *testing* stage, the KB obtained is evaluated to determine whether the current *granularity* and *coverage* of the KB is appropriate for the domain-specific tasks. In the course of building a domain specific KB, revisions may occur.

### 2.2 SYNTAX AND SEMANTICS OF LOGIC-BASED KR LANGUAGES

Logic-based KR languages are defined by giving a syntax and a semantics. In the *syntax*, the formulae which can be formed in the language are defined. Formulae are usually defined recursively, i.e., initially, an alphabet is specified and, then, with the use of syntax rules formulae are defined recursively out of symbols from the alphabet. See §5.1, for example, for the syntax definition of a part of the CLASSIC KR language.

In order to define the meaning, or *semantics*, of formulae which represent domain knowledge, a domain[1] description needs to be constructed. A *domain* is defined by a *universe of discourse*, which is the set of *objects*, and by the *object-tuples* which form the relationships between the objects.

The denotational Tarskian semantics[2] of formulae can be specified in two steps. First, the nonlogical symbols which are used in the formulae are associated with parts of the *domain* of interest $\mathcal{D}$. These associations are specified by an *interpretation function*. Second, the formulae can be evaluated (i.e., the truth value is assigned) in accordance with these associations. Here, we make

---

[1]The domain of interest may consist of one or more consistent concrete or fictional worlds.

[2]Semantics of formulae can be defined in other ways. An alternative way, which is not used in this report, is to state a set of axioms $\Gamma$ and define a formula $\phi$ to be true if and only if this formula is logically implied by this set of axioms, i.e., $\Gamma \models \phi$. We have not chosen this alternative since we want the formulae to apply to the domain of interest.

use of semantic rules for logical symbols within the formulae. A formula is true if and only if it accurately describes the domain.

More formally, the semantics of formulae can be determined by the evaluation of the formulae with respect to a certain *model*. A model $\mathcal{M}$ can be defined as $\mathcal{M} = \langle \mathcal{D}, \mathcal{I} \rangle$ where the *interpretation function* $\mathcal{I}$ is a mapping between the nonlogical symbols of the language and elements of the domain of interest $\mathcal{D}$. See §5.2, for example, for the semantic definition of a part of the CLASSIC KR language.

Genesereth & Nilsson [10] have related the conceptualization and formalization stages of KBS development methods to the syntax and semantics of first-order logic. This has resulted in the following integration. At the conceptualization stage, the conceptual base is equated with the domain description. This conceptual domain description is verbalized in terms of objects and object-tuples. Thus, at the conceptualization stage the relevant objects and object-tuples of the domain of interest are selected. Then, at the formalization stage, the output of the conceptualization phase should be represented in first-order logic. (We propose to possibly use another appropriate logic-based KR language.) In addition, it is possible to clearly specify the interface between the conceptualization stage and the formalization stage by assigning semantics to the formulae at the formalization stage in terms of objects and object-tuples at the conceptualization stage.

## 2.3 EXPRESSING KNOWLEDGE AT THE CONCEPTUALIZATION LEVEL

In evaluating the approach of §2.2, we make two remarks. First, the conceptualization stage and formalization stage have to be demarcated. In §2.1, we have argued that these stages have different objectives. In addition to this argument, the conceptualization stage can now be used to define the semantics of the formulae at the formalization stage. We illustrate this for our application in §6.3.

Second, practical problems arise at the conceptualization stage when it is tried to express the knowledge of a complex domain in terms of objects and object-tuples. We give three examples. (1) Within a realistic domain, it is very hard or even impossible to explicitly enumerate all objects and tuples. (2) In a dynamic domain, like most realistic domains, new objects and tuples are added regularly. (3) It is very difficult to express complex descriptions, such as properties of classes of objects.

We illustrate these problems for the Plinius domain, i.e., ceramic science. (1) Quantitative values are used for describing most properties of a material. It would be impossible or at least very unpractical to list all values that might occur as objects. (2) New materials are reported regularly. In fact, it is one of the main goals of ceramics research to design new materials. (3) All materials have a bending strength, a fracture toughness, a melting point, and so on. It would be very unpractical to express this in the form of tuples consisting of a particular material and the property in question. It would take $n \cdot p$ tuples to do this, with $n$ the number of materials and $p$ the number of properties.

We have resolved these problems by introducing extra building blocks which refine the language consisting of only objects and object-tuples. For the problems described above, the following refinements may be carried through. First, *atomic concepts* are introduced which stand for classes of *objects*. Using atomic concepts, it is possible to implicitly describe infinitely many objects. This means that an object does not have to be contained in the universe of discourse extensionally if an atomic concept exists which describes this object intensionally.

In addition, *concepts* are introduced. A concept can be used to intensionally (or prototypically when not all necessary and sufficient properties are known) describe sets of objects, object-tuples, atomic concepts or concepts (i.e., a concept is defined recursively). Therefore, an extensional

enumeration of all items which satisfy the intensional (or prototypical) specification is not needed. Concepts are defined by a description or structure which consists of the enumeration of the objects, object-tuples, atomic concepts and concepts which play a role together with their interdependencies.

The domain of interest can now, for example, be analysed by constructing a set of atomic concepts and, using this as a basis, more complex concepts can be constructed. For example, concepts may be constructed, which over-generalize a part of the domain of interest. Using these over-generalized concepts, the domain description anticipates on "new" sub-concepts in the domain. We have used this approach in the medical domain [11] and the domain of ceramic science (see §4).

We want to make the remark that the intention here is to show that the construction of a domain description for a realistic domain only in terms of objects and object-tuples is practically impossible. We have given a refinement in order to precisely express a domain description for a realistic domain. Other expressions, such as constraints [12], may also be added for this objective.

## 2.4 SUBDIVISION INTO TERMINOLOGY AND ASSERTIONS

Collecting, defining, structuring and representing knowledge about a complex domain is a difficult task. Brachman & Levesque [13] have proposed to separate the *technical vocabulary* of a domain from the facts in that domain.[3,4]

The *technical vocabulary* consists of descriptions which are atomic or more complex structures forming *concepts*. The descriptions of the concepts of the technical vocabulary are be stored in the *terminological* or *description component*. Structures that are used to express factual knowledge are said to be *assertional*. These assertions of the domain of interest are expressed in terms of the concepts of the terminological component. The assertions of the domain are stored in the *assertional component*.

With this subdivision, the modularity is increased as a result of which one hopes the domain looks more transparent and therefore is easier to understand. Due to this separation, special languages have been developed which aim at representing knowledge of one distinguished component, such as terminological/description logics [15,16].

Although a clear subdivision between terminology and assertions can be made on an abstract level, this is not the case for the implementation of this separation [17]. One of the reasons is that in order to define concepts in the terminological component, assertions are used.

In relating the segmentation introduced in this subsection with the development stages for building an ontology KB for a particular domain, we propose to carry out this segmentation through the conceptualization, formalization and implementation stages.

## 2.5 ONTOLOGY AND ONTOLOGY KB

In the literature [5,18,19], an *ontology* has become known as a systematic vocabulary of concepts and relations of a particular domain. An ontology is language-independent in the sense that a single concept in an ontology as a rule corresponds to different words and phrases, both within a single natural language and in different natural languages. In addition, the interdependencies between the concepts and relations of an ontology need to be specified which lead to a kind of framework. Often these interdependencies are used to define the concepts and relations. Since a systematic vocabulary

---

[3]Within the field of AI, the wish to separate these notions was firstly introduced in [14] in order to clarify the informal semantic networks of that time.

[4]This subdivision corresponds to the separation of a *database scheme* from their occurrences in the database world.

together with a framework are constructed while analysing the domain, and an ontology is made on a conceptual level, we argue that an ontology should be situated within the conceptualization stage.

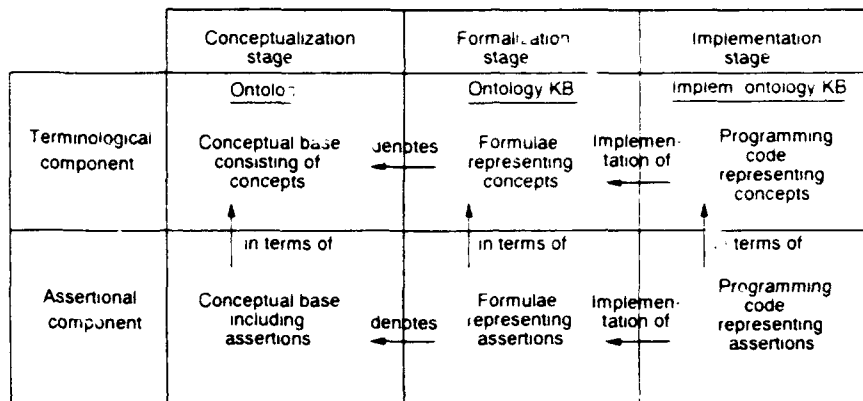| | Conceptualization stage | | Formalization stage | | Implementation stage |
|---|---|---|---|---|---|
| Terminological component | Ontolo ~ | | Ontology KB | | implem ontology KB |
| | Conceptual base consisting of concepts | denotes | Formulae representing concepts | Implementation of | Programming code representing concepts |
| | in terms of | | in terms of | | terms of |
| Assertional component | Conceptual base including assertions | denotes | Formulae representing assertions | Implementation of | Programming code representing assertions |

Figure 1: A development method for building an ontology KB for a particular domain. The conceptualization stage is preceded by the elicitation stage and the implementation stage is followed by the testing stage. The interface between the conceptualization and formalization stage is specified by a semantical, representational relation. The different stages are subdivided into a terminological and an assertional component. The assertions of the assertional component are expressed in terms of concepts of the terminological component.

In the field of knowledge representation (i.e., at the formalization stage), it has been proposed to separate the terminological knowledge from assertional knowledge within a particular domain (§2.4). Since the terminology of a particular domain is expressed within the systematic vocabulary of the domain, we argue that an ontology should be placed within the terminological component. Thus, we argue that an ontology represents the terminological component at the conceptualization stage (see Figure 1). In the remainder of this paper we call the concepts at this stage $C$-concepts. The ontology KB at the formalization stage consists of $\mathcal{F}$-concepts.

The major advantage of incorporating an ontology within the development of a KBS which cover a realistic complex domain (which results in a large KB) is that the domain knowledge is structured according to the framework of the ontology [20]. For a KBS which contains more than one KB, the different KBs are tuned due to the joint use of the same terminology [21,22].

## 3. AN OVERVIEW OF THE PLINIUS PROJECT

In this section, we give a brief description of the Plinius project (for a more detailed description, we refer to [23]). The Plinius project is aimed at developing a system which is able to semi-automatically extract domain-specific knowledge from title and abstract of scientific papers in the field of ceramic science. The course of the documents through the Plinius processes and the knowledge source these processes use are shown in Figure 2.

At the beginning, document descriptions consisting of identifier, title and abstract are prepared for natural language processing. This process of mainly syntactical manipulation is called the Plinius preprocess. The first version of this preprocess has been implemented [24].

Then, in the Plinius language dependent process a syntactic and semantic analysis is carried out on the preprocessed documents which leads to representations of the documents in a formal language. These formal representations are stored in the interim KB. This process (mainly the semantic analysis) is currently under consideration. The Plinius language-dependent process makes
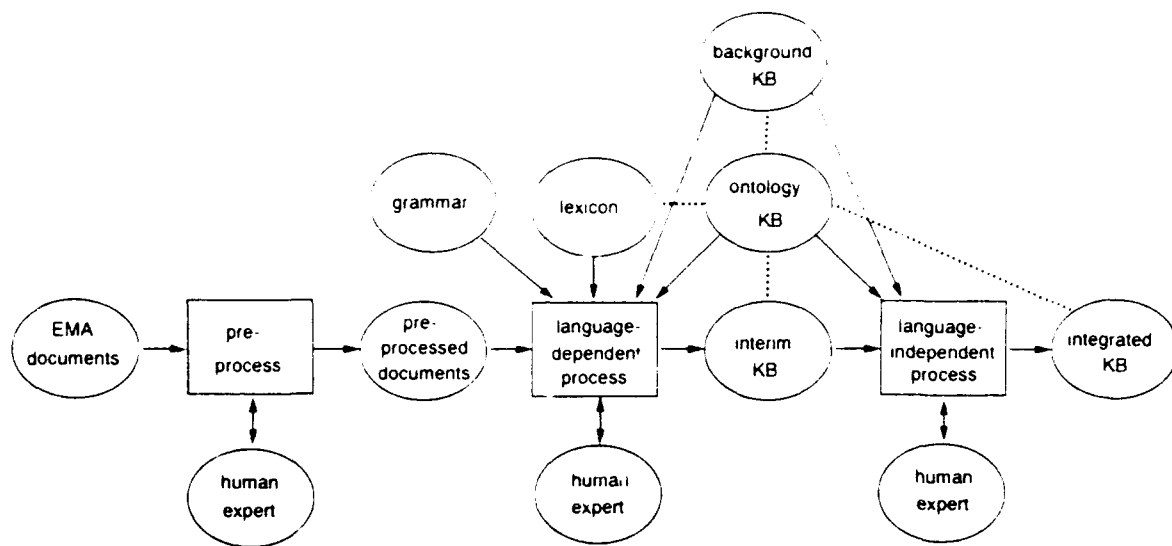
Figure 2: The Plinius processes and knowledge sources.

use of two kinds of knowledge sources. The first kind consists of a sublanguage grammar and a lexicon with syntactical information. These knowledge sources are natural-language (English) dependent.[5] The second kind consists of the background KB, the ontology KB and the part of the lexicon with semantic translations. These knowledge sources are domain dependent and natural-language independent. In addition, the background KB and the semantic part of the lexicon (and the integrated lexicon) contain the concepts and structures of the ontology KB.

Finally, in the Plinius language-independent process the separated representations per document of the interim KB are integrated. The result of this process is stored in the integrated KB (which of course is domain-specific as well). The process makes only use of domain-specific knowledge sources, which include the background KB and the ontology KB. The Plinius language-independent process is being explored.

Plinius does not aim at a fully automatic system. All three processes can communicate with a human operator who deals with ambiguities and other problems the system cannot solve automatically.

The interim KB and the integrated KB are the products of the Plinius project and can be used for several applications. The interim KB might be used in a system for semi-automatic indexing [11]. The integrated KB is meant to be a general purpose domain-specific KB and might be used to answer user queries directly (in the ideal case all user queries which can be answered by reading EMA can also be answered with the integrated KB). In addition, the integrated KB might also be used as KB within domain-specific KBSs.

## 4. AN ONTOLOGY OF CERAMICS: THE PLINIUS ONTOLOGY

The Plinius ontology has been formulated manually using several handbooks of ceramics and materials science. We refer to [25] for a detailed, complete description and motivation. In this section, we only discuss (due to lack of space) the definitions for *properties* of materials and for

---

[5] These knowledge sources are domain dependent since sublanguage is domain dependent.

*processes*. The definition for *material* is treated as a black box[6] (and so does the definition for *phase*, which is needed in the process characterization).

Properties of materials are expressed as quantities in the physical sense, i.e., as comprising a name to identify the kind of quantity involved, a numerical value, a measure of the accuracy (normally the standard deviation), and a unit. A complication arises because certain values are observed under particular conditions. For example, the material's property of displaying an elongation may be qualified by specifying the temperature range. These conditions, to be called *property conditions* here, can be expressed as quantities. Conceptually, property conditions may be called second-order properties or properties of properties.

For the conceptualisation of processes, the so-called *process triangle* is an important notion. A process is fully characterised by three groups of specifications: (1) the input materials; (2) the product; and (3) the conditions under which the process is performed. Theoretically, any of these three can be derived if the other two are known. In practice the necessary theory for performing these derivations is lacking.

The way out proposed in the Plinius ontology is the following. Since theoretically two vertices suffice, the $C$-concept for a process can be defined as comprising two vertices while the third vertex is added but is considered to be non-defining. The choice of the two vertices is arbitrary from a theoretical point of view. Choosing (1) and (2), i.e., the input and output materials, has certain advantages such as allowing easy concatenation of processes. The conditions under which a process is performed are specified by the atmosphere of the process and a set of additional process properties. A process type name, which is a kind of summary of many relevant process conditions, is also added to combat the severe underspecification encountered in the texts.

The part of the Plinius ontology which is needed to introduce the previously described atomic concepts and $C$-concepts contains:[7]

---

The atomic concept $\mathbb{R}$ stands for the class of real numbers.

The atomic concept $\mathbb{Q}$ stands for the class of quantity-names consisting of the objects: grain_size, porosity, bending_strength, size, mass, temperature, duration, tensile stress, frequency, ....

The atomic concept $\mathbb{U}$ stands for the class of measurement units consisting of the objects: meter, kilogram, second, ampere, kelvin, mol, candela, per_second, pascal, meter^3, fraction, ....

The atomic concept $\mathbb{P}$ stands for the class of process type names consisting of the objects: slipcasting, reaction sintering, molten particle deposition, hot isostatic pressing, ....

The $C$-concept Quantity $q$ stands for a tuple

$$q = \langle n, v, d, u \rangle \text{ where } n \in \mathbb{Q}, v, d \in \mathbb{R}, \text{ and } u \in \mathbb{U}$$

where $n$ is a quantity-name, $v$ is a value or value range, $d$ a standard deviation, and $u$ a unit.

In this paper, the $C$-concept Material $m$ is defined as a black box.

The $C$-concept Material property $mp$ stands for the $C$-concept:

$$mp = \langle m, s \rangle$$

---

[6] In [25], *material* is defined as a $C$-concept which is constructed out of several $C$-concepts and atomic concepts.

[7] In this paper, we use the following orthographical conventions: objects are notated with lower case letters in sans serif font (e.g., temperature), atomic concepts by upper case letters in the so-called blackboard bold font, (e.g., $\mathbb{R}$), and $C$-concepts by an upper case letter followed by lower case letters in sans serif font (e.g., Process).

where $m$ stands for a Material and $s$ stands for a set of Quantities.

The $\mathcal{C}$-concept Property condition $pc$ stands for the $\mathcal{C}$-concept:

$$pc = \langle q, s \rangle$$

where $q$ stands for a Quantity expressing a property and $s$ stands for a set of Quantities expressing the conditions.

The $\mathcal{C}$-concept Process $p$ stands for the $\mathcal{C}$-concept:

$$p = \langle m_1, m_2, ptn, ph, s \rangle$$

where $m_1$ stands for the input Material, $m_2$ stands for the output Material, $ptn \in \mathcal{P}$, $ph$ stands for the atmosphere expressed with a Phase (where Phase, similar to material, in this paper is considered as a black box) and $s$ stands for a set of process Quantities.

## 5. THE CLASSIC KNOWLEDGE REPRESENTATION LANGUAGE

In this section, we concentrate on the representation of an ontology at the formalization stage. One group of KR systems, which is suitable for this purpose is the KL-ONE family of KR systems [9]. These KL-ONE style systems can be used very well for the representation of an ontology because of their ability to represent a conceptual vocabulary separately from the assertional knowledge, their ability to define the terms of the conceptual vocabulary intensionally, their logical foundation and their general purpose approach. A descendent of the KL-ONE family of KR systems which is developed at AT&T Bell laboratories is called CLASSIC [26–28]. We have chosen to use the language of this system for the Plinius application since the expressiveness of this language is restricted in order to increase the computational performance. In this section, we give a brief description of the syntax and semantics of a part of the CLASSIC KR language.

### 5.1 SYNTAX OF THE CLASSIC KR LANGUAGE

Following the syntax definition of logic-based KR languages, as discussed in §2.2, we give the syntax of a part of the CLASSIC KR language. Using this syntax the Plinius ontology is represented in §6.2. The alphabet consists of the following sets of symbols:[8]

Nonlogical symbols: subdivided in *individual_name* (which denotes an object), *role_name* (which denotes a class of 2-tuples), and *$\mathcal{F}$-concept_name* (which denotes a class of objects);
Auxiliary symbols: "(" and ")";
Fixed set of function constants: **INDIVIDUAL, AND, ALL, ONE-OF** and **FILLS**; and
The predicate symbol "$\doteq$".

Now, using the sets of symbols of the alphabet, construction rules can be formulated in order to define the sets of *individuals*, *roles* and *$\mathcal{F}$-concepts* (rules 1–8) and formulae are defined (rules 9–10):

| (1) | *individual* | ::= *individual_name* |
| (2) | | \| (**INDIVIDUAL** *$\mathcal{F}$-concept*) |
| (3) | *role* | ::= *role_name* |

---

[8]In this paper, we use the following orthographical conventions: Individual_name: lower case, with first letter uppercase; role_name: lower case; $\mathcal{F}$-CONCEPT_NAME: small caps; FUNCTION_CONSTANT: small caps, boldface font.

| (4) | $\mathcal{F}$-concept | $::=$ $\mathcal{F}$-concept_name |
| (5) | | (AND $\mathcal{F}$-concept$_1$ ... $\mathcal{F}$-concept$_n$), for $n \geq 1$ |
| (6) | | (ALL role $\mathcal{F}$-concept) |
| (7) | | (ONE-OF individual$_1$ ... individual$_n$), for $n \geq 0$ |
| (8) | | (FILLS role individual$_1$ ... individual$_n$), for $n \geq 1$ |
| (9) | formula | $::=$ $\mathcal{F}$-concept_name $\doteq$ $\mathcal{F}$-concept |
| (10) | | individual_name $\doteq$ individual |

## 5.2 SEMANTICS OF THE CLASSIC KR LANGUAGE

In this subsection, the semantics of the previously described parts of the CLASSIC KR language is defined (according to the model discussed in §2.2). In §6.3, we use this semantics to determine the correctness of the Plinius ontology KB formulae with respect to a model including the Plinius ontology.

The interpretation function $\mathcal{I}$ maps the CLASSIC constants role_name on a set of 2-tuples, $\mathcal{F}$-concept_name on a class of objects and individual_name on an object of the universe of discourse. The CLASSIC function constant INDIVIDUAL used in the syntax definition of individual is mapped on an object and the other CLASSIC function constants are mapped on classes of objects. Finally, the model-theoretic semantics of the CLASSIC formulae can be specified. This leads to the following rules (individual and concept are abbreviated):

| (1) | $\mathcal{I}[ind\_name]$ | $\in \mathcal{D}$ |
| (2) | $\mathcal{I}[(\text{INDIVIDUAL } \mathcal{F}\text{-conc})]$ | $\in \mathcal{I}[\mathcal{F}\text{-conc}]$ |
| (3) | $\mathcal{I}[role\_name]$ | $\subseteq \mathcal{D} \times \mathcal{D}$ |
| (4) | $\mathcal{I}[\mathcal{F}\text{-conc\_name}]$ | $\subseteq \mathcal{D}$ |
| (5) | $\mathcal{I}[(\text{AND } \mathcal{F}\text{-conc}_1 \ldots \mathcal{F}\text{-conc}_n)]$ | $= \{x \in \mathcal{D} \mid x \in \mathcal{I}[\mathcal{F}\text{-conc}_1] \wedge \ldots \wedge x \in \mathcal{I}[\mathcal{F}\text{-conc}_n]\}$ |
| (6) | $\mathcal{I}[(\text{ALL } role \, \mathcal{F}\text{-conc})]$ | $= \{x \in \mathcal{C} \mid \forall y \, (\langle x, y \rangle \in \mathcal{I}[role] \Rightarrow y \in \mathcal{I}[\mathcal{F}\text{-conc}])\}$ |
| (7) | $\mathcal{I}[(\text{ONE-OF } ind_1 \ldots ind_n)]$ | $= \{\mathcal{I}[ind_1], \ldots, \mathcal{I}[ind_n]\}$ |
| (8) | $\mathcal{I}[(\text{FILLS } role \, ind_1 \ldots ind_n)]$ | $= \{x \in \mathcal{C} \mid \langle x, \mathcal{I}[ind_1]\rangle \in \mathcal{I}[role] \wedge \ldots$ |
| | | $\wedge \langle x, \mathcal{I}[ind_n]\rangle \in \mathcal{I}[role]\}$ |
| (9) | $\models_{\mathcal{M}} \mathcal{F}\text{-conc\_name} \doteq \mathcal{F}\text{-conc}$ | iff $\mathcal{I}[\mathcal{F}\text{-conc\_name}] = \mathcal{I}[\mathcal{F}\text{-conc}]$ |
| (10) | $\models_{\mathcal{M}} ind\_name \doteq ind$ | iff $\mathcal{I}[ind\_name] = \mathcal{I}[ind]$ |

## 6. THE PLINIUS ONTOLOGY KB IN CLASSIC

The $\mathcal{C}$-concepts of the Plinius ontology are mainly expressed by complex tuples, complex sets, constraints, conjunctions and disjunctions [25]. In §6.1, we show how tuples can be represented in the CLASSIC KR language. Representations for complex sets, constraints, conjunctions and disjunctions are given in [7]. Then, in §6.2, the part of the Plinius ontology as defined in §4 can be represented within in the CLASSIC part as defined in §5.1. Finally, in §6.3, we discuss the semantics of the formulae of the Plinius ontology KB.

## 6.1 THE REPRESENTATION OF (COMPLEX) TUPLES IN CLASSIC

For the representation of $\mathcal{C}$-concepts like Quantity in the CLASSIC KR language, a problem arises. $\mathcal{C}$-concepts sometimes stand for (complex) tuples. In order to represent these (complex) tuples in the CLASSIC KR language, we need constructions which denote sets of tuples. However,

the CLASSIC KR language only contains individuals, $\mathcal{F}$-concepts and roles. An individual denotes an object, an $\mathcal{F}$-concept denotes a set of objects or a set of (atomic) $\mathcal{C}$-concepts, and a role denotes a set of 2-tuples (where the arguments of the 2-tuples may be objects or (atomic) $\mathcal{C}$-concepts). Although a role seems to fulfil the conditions described above, it is not possible to use a role for the representation of (complex) tuples, since a role cannot occur independently in CLASSIC, it only occurs within an $\mathcal{F}$-concept. This means that it is not possible to represent a (complex) tuple directly in the CLASSIC KR language, which is in harmony with [27], page 416:

> "[...] CLASSIC is likely to be cumbersome to use in cases where mathematical entities such as tuples, sequences, geometric entities, etc. are the center of attention."

Schmolze [29] has developed an extension of KR systems like CLASSIC in order to represent (complex) tuples (which he calls $n$-ary relations, with $n > 1$. However, due to the increasing expressiveness, the complexity for reasoning (such as subsumption) decreases. Therefore, we try to represent (complex) tuples in the CLASSIC KR language.

In order to represent (complex) tuples in the CLASSIC KR language, we look at these tuples as $\mathcal{C}$-concepts where each argument of the (complex) tuple is specified by a relation with exactly one filler. This means that (complex) tuples can be represented by $\mathcal{F}$-concepts consisting of an **AND** construction where each conjunct represents an argument and where each conjunct consists of an **ALL** construction where the name of the attribute[9] specifies the argument name and the value restriction specifies what $\mathcal{F}$-concept/individual should be filled in in the argument.

For example, if we look at a simplified definition of the $\mathcal{C}$-concept **Quantity** which stands for a tuple

$$q = \langle n, v, u \rangle \text{ where } n \in \mathbb{Q}, v \in \mathbb{R}, \text{ and } u \in \mathbb{U}$$

we may represent this by the formula

QUANTITY $\doteq$ (**AND** (**ALL** has_quantity_name QUANTITY_NAME)
                    (**ALL** has_value NUMBER)
                    (**ALL** has_unit UNIT))

where the $\mathcal{F}$-concept name QUANTITY represents the $\mathcal{C}$-concept **Quantity** which is defined by an $\mathcal{F}$-concept consisting of an **AND** construction which conjuncts three **ALL** constructions where the roles has_quantity_name, has_value and has_unit specify the argument names quantity name, value and unit respectively, and the value restrictions QUANTITY_NAME, NUMBER and UNIT specify what individual should be filled in in these arguments.

## 6.2 THE SYNTAX OF THE PLINIUS ONTOLOGY KB IN CLASSIC

The CLASSIC representation for the Plinius ontology as described in §4 is the following:

| | |
|---|---|
| QUANTITY_NAME | $\doteq$(**ONE-OF** Grain_size Porosity Bending_strength Size Mass Temperature Duration Tensile_stress Frequency) |
| UNIT | $\doteq$(**ONE-OF** Meter Kilogram Second Ampere Kelvin Mol Candela Per_second Pascal Meter^3 Fraction) |
| PROCESS_TYPE_NAME | $\doteq$(**ONE-OF** Slipcasting Reaction_sintering |

---

[9] An attribute is a role which has at least one and at most one role filler.

| | Molten particle deposition Hot isostatic pressing) |
|---|---|
| QUANTITY | $\doteq$(AND (ALL has quantity name QUANTITY NAME) |
| | (ALL has value NUMBER) |
| | (ALL has standard deviation NUMBER) |
| | (ALL has unit UNIT)) |
| MATERIAL | $\doteq$■ |
| MATERIAL PROPERTY | $\doteq$(AND (ALL has material MATERIAL) |
| | (ALL has property QUANTITY)) |
| PROPERTY CONDITION | $\doteq$(AND (ALL has quantity QUANTITY) |
| | (ALL has condition QUANTITY)) |
| PHASE | $\doteq$■ |
| PROCESS | $\doteq$(AND (ALL has starting material MATERIAL) |
| | (ALL has end material MATERIAL) |
| | (ALL has type PROCESS TYPE NAME) |
| | (ALL has process atmosphere PHASE) |
| | (ALL has property QUANTITY)) |

where all roles except for 'has condition' and 'has property' are defined to have exactly one filler (these roles are attributes) and where the roles 'has condition' and 'has property' are defined to have one or more fillers.

For the representation of the atomic C-concept ℝ, the built-in F-concept name NUMBER is used. For the representation of the atomic C-concepts ℚ, ℂ, and ℙ, respectively the F-concept names QUANTITY NAME, UNIT and PROCESS TYPE NAME are introduced which are defined as sets of individuals. The C-concepts **Property condition**, **Material property** and **Process** are represented by MATERIAL PROPERTY, PROPERTY CONDITION and PROCESS. Finally, the C-concepts **Phase** and **Material** are represented by MATERIAL and PHASE.

## 6.3 THE SEMANTICS OF THE PLINIUS ONTOLOGY KB IN CLASSIC

In this section we discuss the semantics of the formulae of the Plinius ontology KB. Due to these semantics the interface between the conceptualization stage, which resulted in the Plinius ontology, and the formalization stage, which resulted in the Plinius ontology KB, is clarified and the correctness of the formulae is determined with respect to a model which includes the Plinius ontology.

In giving the semantics of the formulae of the Plinius ontology KB, per formula the following phases should be followed. (1) The F-concept name (the left hand side of the formula) needs to be mapped on the corresponding (atomic) C-concept of the Plinius ontology. (2) The F-concept (the right hand side of the formula) which syntactically define the F-concept name, need to be mapped on objects, object-tuples, (atomic) F-concepts of the Plinius ontology. (3) The truth of the formula of the Plinius ontology KB is proven if the denotation of the F-concept name is equal to the denotation of the corresponding F-concept.

We illustrate this by giving the semantics of the QUANTITY-formula. In order to show that this formula has the right meaning with respect to a certain model $\mathcal{M}$, we give the interpretations of the F-concept name and the F-concept and determine whether the formula is correct with respect to this model.

In order to show that the previously given formula has the right meaning with respect to a certain model $\mathcal{M}$, we give the interpretations of the $\mathcal{F}$-concept name and the $\mathcal{F}$-concept and determine whether the formula is correct with respect to this model.

The $\mathcal{F}$-concept name QUANTITY denotes the class of all **Quantities**, i.e.,[10]

$$
\begin{aligned}
\mathcal{I}[\text{QUANTITY}] &= \text{the class of all \textbf{Quantities}} \\
&= \{q \mid q = \langle n, v, u \rangle \wedge n \in \mathcal{Q} \wedge v \in \mathfrak{R} \wedge u \in \mathfrak{U}\} \\
&= \{q \mid q = \langle \text{has quantity name: } n, \text{ has value: } v, \text{ has unit: } u \rangle \\
&\qquad \wedge n \in \mathcal{Q} \wedge v \in \mathfrak{R} \wedge u \in \mathfrak{U}\}
\end{aligned}
$$

The $\mathcal{F}$-concept used to define QUANTITY has the following formal meaning, according to rules 5 and 6 of §5.2:

$$
\begin{aligned}
\mathcal{I}[(\textbf{AND} &(\textbf{ALL} \text{ has\_quantity\_name QUANTITY\_NAME}) \\
&(\textbf{ALL} \text{ has\_value NUMBER }) \\
&(\textbf{ALL} \text{ has\_unit UNIT }))] = \\
&\{q \in \mathcal{D} \mid q \in \mathcal{I}[(\textbf{ALL} \text{ has\_quantity\_name QUANTITY\_NAME})] \\
&\qquad \wedge q \in \mathcal{I}[(\textbf{ALL} \text{ has\_value NUMBER})] \\
&\qquad \wedge q \in \mathcal{I}[(\textbf{ALL} \text{ has\_unit UNIT})]\}
\end{aligned}
$$

where (since has_quantity_name is an attribute)

$$
\begin{aligned}
\mathcal{I}[(\textbf{ALL} \text{ has\_quantity\_name QUANTITY\_NAME})] = \\
\{q \in \mathcal{C} \mid \forall n \, (\langle q, n \rangle \in \mathcal{I}[\text{has\_quantity\_name}] \Rightarrow n \in \mathbb{Q}) \\
\wedge \, card(\{n \in \mathcal{D} \mid \langle q, n \rangle \in \mathcal{I}[\text{has\_quantity\_name}]\}) = 1\}
\end{aligned}
$$

In addition, since (1) attribute has_quantity_name specifies the argument name of the first argument of $q = \langle n, v, u \rangle$ and attributes has_value and has_unit specify the argument names of the second and third argument of $q = \langle n, v, u \rangle$, (2) the first argument of $q = \langle n, v, u \rangle$ may only be filled by a quantity name, the second argument by a number and the third argument by a unit, and (3) the arguments have exactly one role filler, the following equation can be stated:

$$
\begin{aligned}
\mathcal{I}[(\textbf{AND} &(\textbf{ALL} \text{ has\_quantity\_name QUANTITY\_NAME}) \\
&(\textbf{ALL} \text{ has\_value NUMBER }) \\
&(\textbf{ALL} \text{ has\_unit UNIT }))] = \\
&\{q \in \mathcal{C} \mid \forall n \, (\langle q, n \rangle \in \mathcal{I}[\text{has\_quantity\_name}] \Rightarrow n \in \mathbb{Q}) \\
&\qquad \wedge \, card(\{n \in \mathcal{D} \mid \langle q, n \rangle \in \mathcal{I}[\text{has\_quantity\_name}]\}) = 1 \\
&\quad \wedge \, \forall v \, (\langle q, v \rangle \in \mathcal{I}[\text{has\_value}] \Rightarrow v \in \mathbb{R}) \\
&\qquad \wedge \, card(\{v \in \mathcal{D} \mid \langle q, v \rangle \in \mathcal{I}[\text{has\_value}]\}) = 1 \\
&\quad \wedge \, \forall u \, (\langle q, u \rangle \in \mathcal{I}[\text{has\_unit}] \Rightarrow u \in \mathbb{U}) \\
&\qquad \wedge \, card(\{u \in \mathcal{D} \mid \langle q, u \rangle \in \mathcal{I}[\text{has\_unit}]\}) = 1\} =
\end{aligned}
$$

$$
\begin{aligned}
\{q \in \mathcal{C} \mid q = \langle \mathcal{I}[\text{has\_quantity\_name}] : n, \; \mathcal{I}[\text{has\_value}] : v, \; \mathcal{I}[\text{has\_unit}] : u \rangle \\
\wedge \, n \in \mathbb{Q} \wedge v \in \mathbb{R} \wedge u \in \mathbb{U}\} =
\end{aligned}
$$

$$
\{q \mid q = \langle n, v, u \rangle \wedge n \in \mathbb{Q} \wedge v \in \mathbb{R} \wedge u \in \mathbb{U}\}
$$

and thus we conclude that the denotation of $\mathcal{F}$-concept name QUANTITY is equal to the denotation of its defining $\mathcal{F}$-concept. This means, by rule 9 of §5.2:

---

[10]Since the argument names of the tuple $\langle n, v, u \rangle$ are not specified within the notation, we introduce the alternative notation $\langle$ has quantity name: $n$, has value: $v$, has unit: $u \rangle$ in the last line.

$\models_{\mathcal{M}}$ QUANTITY $\doteq$ (**AND** (**ALL** has_quantity_name QUANTITY_NAME)
(**ALL** has_value NUMBER)
(**ALL** has_unit UNIT))

Thus, the QUANTITY formula is correct with respect to model $\mathcal{M}$ where $\mathcal{D}$ is the Plinius ontology and $\mathcal{I}$ the interpretations given above.

In [7], the semantics of all Plinius formulae are given by specifying a complete model which includes the Plinius ontology and the interpretation functions for all $\mathcal{F}$-concept names, $\mathcal{F}$-concepts, roles, individual names and individuals.

## 7. CONCLUSIONS

Using and integrating existing AI subjects, we have constructed a development method for building and implementing an ontology KB for a particular domain. This method consists of the elicitation, conceptualization, formalization, implementation, and testing stages. The formalization stage is coupled with a logic-based KR language, the conceptualization stage is coupled with a domain description and the interface between the conceptualization and formalization stage is specified as a semantical, representational relation. This means that the conceptualization stage is used for both the analysis of the domain of interest and the specification of the meaning (semantics) of the symbols at the formalization stage. This is in contrast to the approach used by applications for the KL-ONE style KR languages where the conceptualization and formalization stages have not been distinguished. Finally, the conceptualization, formalization, and implementation stages are subdivided into a terminological and an assertional component. An ontology represents the terminological component at the conceptualization stage.

This proposed method has been applied to the Plinius project for which an ontology for ceramic science had been developed [25]. This ontology, which makes use of tuple, set and disjunction expressions, has been formalized and represented within the CLASSIC KR language. For the representation of complex tuples and complex sets, special complex constructions have been developed within CLASSIC. Disjunctions can only be represented under certain conditions. This means that it is possible to represent the Plinius ontology in CLASSIC, however, the restricted expressiveness takes its toll.

In contrast to most represented KBs, we have explicitly specified the semantics of the formulae. Due to this semantics, the correctness of the formulae with respect to the ontology is guaranteed. This means that the task of constructing a formalized ontology can be formally divided into two subtasks, (1) the design of an ontology, independent of representational aspects, and (2) the representation of the ontology, independent of concerns whether a right ontology of the domain of interest has been developed. Although empirical justifications are absent, the method as described in this paper seems to be useful, at least for the different domain specific KBs within the Plinius project.

## 8. ACKNOWLEDGEMENTS

## 9. REFERENCES

[1] Otthein Herzog & Claus-Rainer Rollinger, *Text understanding in LILOG: integrating computational linguistics and artificial intelligence, final report on the IBM Germany LILOG-Project*. Lecture notes in artificial intelligence; subseries of lecture notes in computer science; edited by Jörg Siekmann; volume 546, Springer-Verlag, Berlin, 1991.

[2] Betsy L. Humphreys, Donald A.B. Lindberg & William T. Hole, "Assessing and enhancing the value of the UMLS Knowledge Sources," in *Proceedings of the Fifteenth Annual Symposium on Computer Applications in Medical Care, a conference of the American Medical Informatics Association, Washington, DC, November 17–20, 1991*, Paul D. Clayton, ed., McGraw-Hill, New York, NY, 1991, 194–198.

[3] Douglas B. Lenat & Ramanathan V. Guha, *Building large knowledge-based systems: representation and inference in the Cyc Project*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1990.

[4] Gudrun Klose & Thomas Pirlein, "Modelling knowledge for a natural language understanding system," in *Proceedings of the Fifth European Chapter of the Association for Computational Linguistics*, 1991.

[5] Thomas R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," Knowledge Systems Laboratory, Stanford University, Technical Report KSL 93–4, Palo Alto, CA, 1993, Submitted to AAAI'93.

[6] Nicola Guarino, "Concepts, attributes and arbitrary relations: some linguistic and ontological criteria for structuring knowledge bases," *Data & Knowl. Eng.* 8 (1992), 249–261.

[7] Piet-Hein Speel, "The Plinius ontology knowledge base in CLASSIC," University of Twente, Memorandum UT-KBS-93-08, Enschede, the Netherlands, 1993.

[8] Frederick Hayes-Roth, Donald A. Waterman & Douglas B. Lenat, *Building expert systems*, Addison-Wesley, Reading, MA, 1983.

[9] William A. Woods & James G. Schmolze, "The KL-ONE family," in *Semantic networks in artificial intelligence*, Fritz Lehmann, ed., Modern applied mathematics and computer science; edited by Ervin Y. Rodin; Volume 24, Pergamon Press, Oxford, England, 1992, 133–177, Published as a special issue of the journal *Computers & Mathematics with Applications 23*.

[10] Michael R. Genesereth & Nils J. Nilsson, *Logical foundations of artificial intelligence*, Morgan Kaufmann Publishers, Inc., Palo Alto, 1987.

[11] Piet-Hein Speel, Nicolaas J.I. Mars & Paul E. van der Vet, "A knowledge-based approach to semi-automatic indexing," in *Proceedings of the Workshop on Language & Information Processing, October 27, 1991, Washington, DC, held at the 54th ASIS Annual Meeting*, Alexa T. McCray, ed., 1991, 49–58.

[12] John F. Sowa, "Conceptual analysis as a basis for knowledge acquisition," in *The Cognition of Experts: Psychological Research and Empirical AI*, Springer-Verlag, Berlin, 1991.

[13] Ronald J. Brachman & Hector J. Levesque, "Competence in knowledge representation," in *Proceedings National Conference on Artificial Intelligence, Pittsburgh, PE, 18–20 August 1982*, American Association for Artificial Intelligence, 1982, 189–192.

[14] William A. Woods, "What's in a link: foundations for semantic networks," in *Representation and understanding: studies in cognitive science*, Daniel G. Bobrow & Allan M. Collins, eds., Academic Press, New York, NY, 1975, 35–82.

[15] Franz Baader, Hans-Jürgen Bürckert, Jochen Heinsohn, Bernhard Hollunder, Jürgen Müller, Bernhard Nebel, Werner Nutt & Hans-Jürgen Profitlich, "Terminological knowledge representation: a proposal for a terminological logic," Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI, DFKI note, Saarbrücken, Germany, 1991.

[16] Peter F. Patel-Schneider & William R. Swartout, "Working version: description logic specification from the KRSS effort," 1992.

[17] Robert M. MacGregor, "The evolving technology of classification-based knowledge representation systems," in *Principles of semantic networks: explorations in the representation of knowledge*, John F. Sowa, ed., The Morgan Kaufmann series in representation and reasoning, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991, 385–400.

[18] Robert Neches, Richard E. Fikes, Tim Finin, Thomas Gruber, Ramesh Patil, Ted Senator & William R. Swartout, "Enabling technology for knowledge sharing," *AI Magazine* 12 (1991), 36–56.

[19] Patrick J. Hayes, "Naive physics I: ontology for liquids," in *Formal theories of the commonsense world*, Jerry R. Hobbs & Robert C. Moore, eds., Ablex, Norwood, NJ, 1985, 71–107.

[20] John F. Sowa, "Crystallizing theories out of knowledge soup," in *Intelligent systems: state of the art and future directions*, Zbigniew W. Ras & Maria Zemankova, eds., Ellis Horwood Ltd., London, 1990, 456–487.

[21] Victor Raskin, "Ontology, sublanguage, and semantic networks in natural language processing," in *Advances in Artificial Intelligence: natural language and knowledge based systems*, Martin Charles Golumbic, ed., Springer-Verlag, New York, NY, 1990, 114–128.

[22] Thomas R. Gruber, "The role of common ontology in achieving sharable, reusable knowledge bases," in *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91), Cambridge, MA, April 22-25, 1991*, James Allen, Richard E. Fikes & Erik Sandewall, eds., 1991, 601–602.

[23] Nicolaas J.I. Mars & Paul E. van der Vet, "A semi-automatically generated knowledge base for direct answers to user questions," in *TKE'90: Terminology and knowledge engineering. Proceedings Second International Congress on Terminology and Knowledge Engineering, Trier, 2–4 October 1990*, H. Czap & W. Nedobity, eds., Indeks Verlag, Frankfurt am Main, 1990, 352–362.

[24] Frank van Raalte, Piet-Hein Speel & Paul E. van der Vet, "The Plinius preprocess: intermediate report," University of Twente, Memorandum UT-KBS-92-26, Enschede, The Netherlands, 1992.

[25] Paul E. van der Vet & Nicolaas J.I. Mars, "An ontology of ceramics," University of Twente, UT-KBS-91-21, Memoranda Informatica 91-85, Enschede, The Netherlands, 1991.

[26] Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness & Lori Alperin Resnick, "CLASSIC: a structural data model for objects," in *Proceedings of ACM-SIGMOD 1989 International Conference on Management of Data, Portland, OR, May 31–June 2, 1989*, James Clifford, Bruce Lindsay & David Maier, eds., ACM Press, New York, NY, 1989, 58–67, (also appeared as ACM SIGMOD Record, 18, 2, June, 1989).

[27] Ronald J. Brachman, Deborah L. McGuinness, Peter F. Patel-Schneider, Lori Alperin Resnick & Alexander Borgida, "Living with CLASSIC: when and who to use a KL-ONE-like language," in *Principles of semantic networks: explorations in the representation of knowledge*, John F. Sowa, ed., The Morgan Kaufmann series in representation and reasoning, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1991, 401–456.

[28] Peter F. Patel-Schneider, Deborah L. McGuinness, Ronald J. Brachman, Lori Alperin Resnick & Alexander Borgida, "The CLASSIC knowledge representation system: guiding principles and implementation rationale," *SIGART Bulletin* 2 (1991), 108–113, Special issue on implemented knowledge representation and reasoning systems.

[29] James G. Schmolze, "Terminological knowledge representation systems supporting n-ary terms," in *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR'89), Toronto, Ontario, Canada, May 15-18, 1989*, Ronald J. Brachman, Hector J. Levesque & Raymond Reiter, eds., 1989, 432–443.

# LINGUISTIC TOOLS
# FOR INTELLIGENT SYSTEMS

**Boris Stilman**

Department of Computer Science & Engineering
University of Colorado at Denver,
Campus Box 109, Denver, CO 80217-3364, USA
E-mail: bstilman@copper.denver.colorado.edu

## ABSTRACT

The objective of the research considered in this paper is to develop formal linguistic tools for the representation of large-scale hierarchical complex systems, the so-called Linguistic Geometry. The research relies on the formalization of heuristics of high-skilled human experts which have resulted in the development of successful decision support systems. This approach is based on a broad application of the theory of formal languages and grammars as well as theories of formal problem-solving and planning on the basis of the first-order predicate calculus. This paper reports new results in the investigation of *geometrical properties* of the first-level subsystems (paths of elements) unified as One-Dimensional Linguistic Geometry.

## 1. INTRODUCTION

Many important practical problems can be considered as optimization problems for complex systems. The difficulties we meet trying to find optimal operation for real-world complex systems are well known. While the formalization of the problem, as a rule, is not difficult, an algorithm that finds its solution usually results in the search of many variations. For small-dimensional "toy" problems a solution can be obtained; however, for most real-world problems the dimension increases and the number of variations increases significantly, sometimes exponentially, as a function of dimension [1]. Thus most real-world search problems are not solvable employing exact algorithms in a reasonable amount of time.

A development of approximate algorithms for such problems is a necessity. There have been many attempts to design different approximate algorithms. One of the basic ideas is to decrease the dimension of the real-world system following the approach of a human expert in a certain field, by breaking this system down into subsystems, to study these subsystems separately or in combinations, making appropriate searches, and eventually combining optimal solutions for the subsystems as an approximately optimal solution for the whole system [2–4]. These ideas have been implemented for many problems with varying degrees of success, but each implementation was unique. There was no general approach for such implementations. Each new problem must be carefully studied and previous experience usually can not be applied. On the other hand, every attempt to evaluate the computational complexity and quality of a pilot solution requires implementing its program, which in itself is a unique task for each problem.

Here we consider a formal, general approach for a certain class of search problems that involves breaking down a system into dynamic subsystems. This approach *does not*

*immediately give* us powerful tools for reducing the search in different complex problems. It *does give* us a set of tools to be used for the formal description of problems where successful results have already been achieved due to the informal, plausible reasoning of some human expert. This reasoning should involve the decomposition of a complex system into a hierarchy of dynamic interacting subsystems. The proposed approach permits us to study the secondary multi-level system formally, evaluate the complexity and quality of solutions, improve them, if necessary, and generate computer programs for applications. This approach provides us with an opportunity to transfer formal properties and constructions discovered in one problem to a new one and to apply the same tools to the new problem domain. It actually looks like an application of the methods of a chess expert to a maintenance scheduling problem and vice versa. But what about guaranties of success? The guaranties reside in deeper studies of these methods, in the discovery of inner properties which brought us to a success in a certain class of complex systems.

The main idea of the approach considered in this paper is as follows. A set of dynamic subsystems might be represented as a hierarchy of formal languages where each "sentence" (a group of "words" or symbols) of the lower level language corresponds to the "word" of the higher level one. This is a routine procedure in our native language. For example, the phrase "A man who teaches students" creates a hierarchy of languages. A lower level language is a native language without the word "professor." The symbols of this language are all the English words (except "professor"). A higher level language might be the same language with one extra word "A-man-who-teaches-students". Instead, we can use the word "professor" which is simply a short designation of this long word.

In the 1960's a formal syntactic approach to the investigation of properties of natural language resulted in the fast development of a theory of formal languages by Chomsky [5], Ginsburg [6], and others [7, 8]. This development provided an interesting opportunity for dissemination of this approach to different areas. We refer to the ideas of syntactic methods of pattern recognition developed by Fu [9, 10], Narasimhan [11], and Pavlidis [12], and picture description languages by Shaw [13], Feder [14], and Phaltz [15]. We have transformed the idea of linguistic representation of complex real-world and artificial images into the idea of similar representation of complex hierarchical systems. However, the appropriate languages should possess more sophisticated attributes than languages usually used for pattern description. They should describe mathematically all of the essential syntactic and semantic features of the system and search and be easily generated by certain controlled grammars. An origin of such languages can be traced back to the origin of SNOBOL-4 and the research on programmed formal grammars and languages by Knuth [7], Rozenkrantz [8], and Volchenkov [16]. A mathematical environment for the formal implementation of this approach was developed following the theories of formal problem solving and planning by Nilsson, Fikes [17], Sacerdoti [18], and McCarthy, Hayes [19] on the basis of the first order predicate calculus. To show the power of this approach it is important that the chosen model of the hierarchical system be sufficiently complex, poorly formalized, and has successful applications in different areas. The chosen informal model was developed and applied to scheduling, planning, and computer chess by Botvinnik, Stilman, and others [4].

An application of the hierarchy of languages to the chess model was implemented in full as program PIONEER [4]. For power equipment maintenance the hierarchy was implemented in a number of computer programs being used for maintenance scheduling in the USSR [21, 22].

## 2. EXPERIMENTAL RESULTS

In order justify the following theoretical results we present here a brief discussion

about search algorithms and applications of the considered approach. We look for approximate algorithms that reduce B, the branching factor [20], especially, those algorithms which make B close to 1. Such algorithms should be considered as extremely goal-driven with *minimal branching to different directions*.

Different search algorithms were designed in order to reduce the branching factor. They are dynamic programming, various types of branch-and-bound algorithms, etc. For opposing games like chess the most popular algorithms are various search algorithms with alpha-beta pruning [20]. They are implemented in the most powerful computer chess programs, e.g., in all the programs which are current and former World Computer Chess Champions. It was proved that these algorithms, in the best case, theoretically can reduce the branching factor to $B^{0.5}$ [20]. Supposing that an arbitrary chess position in average contains about 40 moves permitted according to the chess rules, alpha-beta pruning can reduce this number to approximately 6. Still we have an exponential growth with a very *high base* (high branching factor). Thus chess problems that require a deep search, e.g., the search to the depth of 20 or more plies, would require enormous amounts of processing time to be solved. We encounter the same problem but in a greater scope when we apply search algorithms with alpha-beta pruning (or branch-and-bound algorithms) to real-world problems, e.g., when we look for an optimal operation of complex systems. In such problems the number of possibilities in each state usually is far more than 40, so an alpha-beta or branch-and-bound reduction of the branching factor does not provide a solution in a reasonable processing time.

Returning to the discussion of experiments with the PIONEER chess program, let us consider the values of branching factor as well as some other parameters of the search [4]. The search tree generated by PIONEER while solving the R. Reti endgame contained 54 nodes ($T = 54$), hence, taking into account that the length of the solution $L = 6$ here, we have $B \sim 1.65$. In the Botvinnik-Kaminer endgame the total number of nodes generated by the program was equal to 145, maximum length $L = 12$, hence $B \sim 1.34$. Although both endgames are solvable by conventional chess programs, these results are very interesting in the framework of substantial reduction of the branching factor.

Among the variety of complex problems solved by the PIONEER, we shall consider two. Both *are not solved yet* by the conventional chess programs: alpha-beta pruning failed to provide a substantial reduction of the branching factor, and so the expected processing time would exceed a reasonable amount.

The first problem is the G.Nadareishvili endgame [4]. The total number of nodes generated was $T = 200$, while the depth of the search required to find a solution is equal to 25! Consequently, $B \sim 1.14$. At the initial position of this endgame there are 10 pieces, and the unreduced branching factor might be estimated as $B \sim 15$. The second complex problem we would like to consider is the middle-game position in a game by Botvinnik-Capablanca. This position contains 19 pieces and the unreduced branching factor might be estimated as $B \sim 20$! The depth of the search should not be less than 23. The PIONEER generated a search tree of 40 nodes with the branching factor $B \sim 1.05$.

Let us consider experiments with maintenance scheduling programs. The program for *monthly* scheduling generated different search trees depending on the number of demands in each month and a list of other constraints [21, 22]. The number of demands varied from 118 to 405 in different months. The total number of nodes never exceeded 165. With 31 as the maximum length of the solution, a reduced branching factor in these problems never exceeded 1.06. (To understand these results we should take into account that the program aggregated some of the demands. In spite of this the unreduced branching factor varied from 50 to 100.)

The experiments with the program for *annual* maintenance scheduling showed that even this higher dimensional problem can be solved employing the proposed approach. The

power equipment maintenance plan for the USSR United Power System was computed for 1121 demands. Each demand contained 12 parameters, including resources requirements and different types of constraints. Two types of resources were considered: the power reserve and the maintenance personnel. The last one was broken into different specialties. Obviously for the annual plan the length of the solution was 365! The reduced branching factor never exceeded 1.005.

Evaluation of the quality of a solution for the chess problems is not hard. The variant-solution (or subtree) is known. A computer should find it and prove it is optimum. For maintenance scheduling problems the optimal plan is unknown but the results achieved can be evaluated according to the optimum criterion: maximum total demanded power of the units being actually maintained. For monthly scheduling the total demanded power of the solutions varied from 91% to 99% of the theoretical optimum value. For annual scheduling the total demanded power of the solutions was equal to 83% of the total demand while a theoretical optimum was unknown.

The comparison with analogous scheduling programs based on branch-and-bound (or dynamic programming) search strategies showed the advantage of our approach for monthly planning; the quality of the plan was about the same, but the computation time in our case was essentially shorter. In all experiments the branching factor of the trees generated by conventional programs was substantially higher. For yearly planning problems the competition could not even happen, because conventional programs could not overcome in a reasonable time the "combinatorial explosion" for such a higher-dimensional problem.

The results shown by these programs in solving complex chess and scheduling problems indicate that implementations of the hierarchy of languages resulted in the extremely goal-driven algorithms generating search trees with a branching factor close to 1. In order to discover the inner properties of human expert heuristics, which were successful in a certain class of complex systems, we develop a formal theory, the Linguistic Geometry [21–28].

## 3. INFORMAL REVIEW

The idea of a hierarchy of formal languages has been implemented in full for the problems which can be stated as problems of optimal functioning of a Complex System, a twin-set of *elements* and *points* where elements are units moving from one point to another. The elements are divided into two opposite sides: the goal of each side is to maximize a gain, the total value of opposite elements withdrawn from the system. Such a withdrawal happens if an element comes to the point where there is already an element of the opposite side: in this case opposite element should be withdrawn, e.g., as in the game of chess.

According to [16], a one-goal, one-level system should be substituted for a multi-goal multi-level system by introducing intermediate goals and breaking it down into subsystems striving to attain these goals. The goals of the subsystems are individual but coordinated with the main mutual goal. Each subsystem includes elements of both sides; the goal of one side is to attack and gain some element (a target), while the other side tries to protect it. Thus, a subsystem called a Zone is the set of elements of both sides with their trajectories (paths). The pruning criteria for the search and evaluation function are coordinated with the intermediate subsystem's goals and the main goal of the system. Obviously, problems studied in [16] are not the only class of problems eligible for creating a hierarchy of formal languages.

Let us review the linguistic representation. Lower level subsystems are called the *trajectories* of points for moving elements along these points to achieve certain local goals. Trajectories are strings of a lower level formal language, the Language of Trajectories.

Higher level subsystems are well-organized *networks* of trajectories for moving elements along them to achieve cooperative goals, specific for each network. These networks, called Zones, are represented as strings of a yet higher level language, the Language of Zones; each symbol of the string represents a trajectory, i.e., the string of a lower level language.

The system functions by moving from one state to another; that is, the movement of an element from one point to another causes an adjustment of the hierarchy of languages. This adjustment can be represented as a mapping (*translating*) to some other hierarchy. Thus, the functioning of the system, in a process of the search, generates a *tree* of translations of the hierarchy of languages. This tree is represented as a string of the highest level formal language, the Language of Translations, which itself is a member of the family of languages corresponding to various well-known search algorithms: depth-first search, breadth-first search, alpha-beta and others. Every string of the Language of Translations (corresponding to some search tree) contains a solution to the specific search problem.

Next we consider a formal theory and report some results for the lower level subsystems, the so-called trajectories.

## 4. COMPLEX SYSTEMS

A **Complex System** is the following eight-tuple:
$$< X, P, R_p, \{ON\}, v, S_i, S_t, TR>,$$
where $X=\{x_i\}$ is a finite set of *points*; $P=\{p_i\}$ is a finite set of *elements*; P is a union of two nonintersecting subsets $P_1$ and $P_2$; $R_p(x,y)$ is a set of binary relations of *reachability* in X ( x and y of X, p of P); $ON(p)=x$, where ON is a partial function of *placement* from P into X; v is a function on P with positive integer values; it describes the *values* of elements; The Complex System searches a space of states, hence, it should have initial and target states. $S_i$ and $S_t$ are the descriptions of the *initial* and *target* states in the language of the first order predicate calculus, that matches with each relation a certain Well-Formed Formula (WFF). Thus, each state from $S_i$ or $S_t$ is described by a certain collection of WFF of the form $\{ON(p_j)=x_k\}$; TR is a set of operators TRANSITION(p, x, y) of transition of the System from one state to another one. These operators describe the transition in terms of two lists of WFF (to be removed and added to the description of the state), and of WFF of applicability of the transition. Here,

**Remove list:** ON(p)=x, ON(q)=y;

**Add list:** ON(p)=y;

**Applicability:** $(ON(p)=x)^\wedge R_p(x,y)$,

where p belongs to $P_1$ and q belongs to $P_2$ or vice versa. The transitions are carried out in turn with participation of elements p from $P_1$ and $P_2$ respectively; omission of a turn is permitted.

According to definition of the set P, the elements of the System are divided into two subsets $P_1$ and $P_2$. They might be considered as units moving along the reachable points. Element p can move from point x to point y if these points are reachable, i.e., $R_p(x,y)$ holds. The current location of each element is described by the equation ON(p)=x. Thus, the description of each state of the System $\{ON(p_j)=x_k\}$ is the set of descriptions of the locations of the elements. The operator TRANSITION(p, x, y) describes the change of the state of the System caused by the move of the element p from the point x to the point y. The element q from the point y must be withdrawn (eliminated) if p and q belong to the different subsets $P_1$ and $P_2$.

The problem of the optimal operation of the System is considered as a search for the optimal variant of transitions leading from one of the initial states to a target state.

With such a problem statement for search for the optimal sequence of transitions into the target state, we could use formal methods like those in the problem-solving system STRIPS [17], nonlinear planner NOAH [18], or in subsequent planning systems. However the search would have to be made in a space of a huge dimension (for nontrivial examples), i.e., in practice no solution would be obtained. We, thus, devote ourselves to search for an approximate solution of a reformulated problem considering our Complex System in some sense as *nearly decomposable* [2].

It is easy to show that positional games such as chess and checkers might be considered as Complex Systems [21, 24-27]. But it is more interesting that this specific model of the formal linguistic approach is applicable to representing and solving a wide class of practical problems such as power maintenance scheduling, long-range planning, operations planning, VLSI layout, and various operations research problems [21, 22, 26, 27]. The idea is that the optimal variant of operation of these real-world systems may be artificially reduced to a two-sides game where one side strives to achieve some goal and the other is responsible for the provision of resources.

## 5. GEOMETRICAL PROPERTIES OF THE COMPLEX SYSTEM

To create and study a hierarchy of dynamic subsystems we have to investigate geometrical properties of the Complex System.

Fig. 1. An interpretation of the family of reachability areas
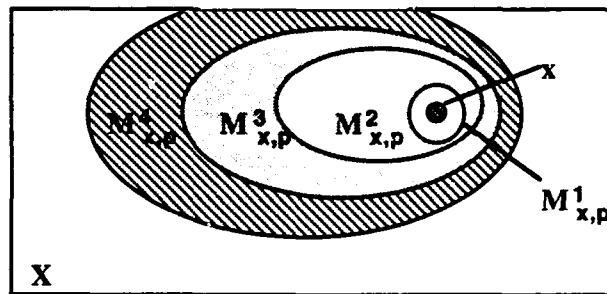


A **map of the set X** relative to the point x and element p for the Complex System is the mapping:

$$MAP_{x,p}: X \longrightarrow Z_+,$$

(where x is from X, p is from P), which is constructed as follows. We consider a *family of reachability areas* from the point x, i.e., a finite set of the following nonempty subsets of X $\{M^k_{x,p}\}$ (Fig. 1):

$k=1$: $M^k_{x,p}$ is a set of points *m reachable in one step* from x: $R_p(x,m)=T$;

$k>1$: $M^k_{x,p}$ is a set of points *reachable in k steps and not reachable in k-1 steps*, i.e., points m reachable from points of $M^{k-1}_{x,p}$ and not included in any $M^i_{x,p}$ with numbers i less than k.

Let $MAP_{x,p}(y)=k$, for y from $M^k_{x,p}$ (*number of steps from x to y*).

In the remainder points let

$MAP_{x,p}(y)=2n$, if $y\neq x$, and

$MAP_{x,p}(y)=0$, if $y=x$.

It is easy to verify that the map of the set X for the specified element p from P defines an *asymmetric distance function* on X:

1. $MAP_{x,p}(y) > 0$ for $x \neq y$;  $MAP_{x,p}(x)=0$;
2. $MAP_{x,p}(y)+MAP_{y,p}(z) \geq MAP_{x,p}(z)$.

If $R_p$ is a symmetric relation,

3. $MAP_{x,p}(y)=MAP_{y,p}(x)$,

In this case each of the elements p from P specifies on X its *own metric*.

## 6. CONTROLLED GRAMMARS

In pattern recognition problems, a linguistic approach was proposed [9-15] for representation of hierarchic structured information contained by each pattern, i.e., for describing patterns by means of simpler subpatterns. This approach brings to light an analogy between the hierarchic structure of patterns and the syntax of languages. The rules controlling the merging of subpatterns into patterns are usually given by the so-called pattern description grammars, with the power of such description being explained by the recursive nature of the grammars. Using similar approach for generating of the hierarchy of formal languages, we make use of the theory of formal grammars in the form developed in [7, 8, 16]. We begin with the definition of the class of grammars to be used.

A **controlled grammar G** is the following eight-tuple:

$$G=(V_T, V_N, V_{PR}, E, H, Parm, L, R),$$

where

$V_T$    is the alphabet of *terminal symbols*;

$V_N$    is the alphabet of *nonterminal symbols*, $S$ (from $V_N$) is the start symbol;

$V_{PR}$  is the alphabet of the *first order predicate calculus PR*:

$V_{PR}=Truth$ U*Con* U*Var* U*Func* U*Pred* U{symbols of logical operations},where

   *Truth* are truth symbols T and F (these are reserved symbols);

   *Con* are constant symbols;

   *Var* are variable symbols;

   *Func* are functional symbols (*Func =Fcon* U*Fvar* ). Functions have an attached non-negative integer referred to as *arity* indicating the number of elements of the domain mapped onto each element of the range. A term is either a constant, variable or function expression. A *function expression* is given by a functional symbol of arity $k$, followed by $k$ terms, $t_1, t_2,..., t_k$, enclosed in parentheses and separated by commas;

   *Pred* are predicate symbols. Predicates have an associated positive integer referred to as *arity* or "argument number" for the predicate. Predicates with the same name but different arities are considered distinct. An *atom* is a predicate constant of arity $n$, followed by $n$ terms, $t_1, t_2,..., t_n$, enclosed in parentheses and separated by commas. The truth values, T and F, are also atoms. *Well-formed formulas* (or WFF) are atoms and combinations of atoms using logical operations;

$E$ is an enumerable set called the *subject domain*;

$H$ is an *interpretation* of *PR* calculus on the set $E$, i.e., a certain assignment of the following form. Each

   -- constant from *Con* is assigned to an element of $E$;

   -- variable from *Var* is assigned to a nonempty subset of $E$; these are allowable substitutions for that variable;

   -- predicate $Q$ from *Pred* of arity $n$ is assigned to a relation on the set $E$ of arity $n$, i.e., to a mapping from $E^n$ into {T, F};

– function $f$ of arity $k$ is assigned to a mapping $h(f)$ from $D$ into $E$, where $D$ belongs to $E^k$. If $f$ is from $Fvar$, then $D$ and the mapping $h(f)$ vary in the process of derivation in the grammar.

Thus, the interpretation $H$ allows us to calculate the value of any function (it lies in $E$) and any predicate (F or T ), if the values of all variables contained by them are specified.

**Parm** is a mapping from $V_T U V_N$ in $2^{Var}$ matching with each symbol of the alphabet $V_T U V_N$ a set of formal parameters, with $Parm(S)=Var$;

**L** is a finite set called the set of *labels*;

**R** is a finite set of *productions*, i.e., a finite set of the following seven-tuples:
$$(l, \ Q, \ A \longrightarrow L, \ \pi_k, \ \pi_n, \ F_T, \ F_F).$$

Here, $l$ (from $L$ ) is the label of a production; the labels of different productions are different, and subsequently sets of labels will be made identical to the sets of productions labeled by them;

$Q$ is a WFF of the predicate calculus $PR$ , the *condition* of applicability of productions; $Q$ contains only variables from $Var$ which belong to $Parm(A)$;

$A \longrightarrow B$ is an expression called the *kernel of production*, where $A$ is from $V_N$; $B$ is from $(V_T U V_N)^*$ is a string in the alphabet of the grammar G;

$\pi_k$ is a sequence of functional formulas corresponding to all formal parameters of each entry of symbols from $V_T U V_N$ into the strings $A$ and $B$ (*kernel actual parameters*);

$\pi_n$ is a sequence of functional formulas corresponding to all formal parameters of each functional symbol from $Fvar$ (*non-kernel actual parameters*);

$F_T$ is a subset of $L$ of labels of the productions permitted on the next step of derivation if $Q=T$ ("true"); it is called a *permissible set in case of success*;

$F_F$ is a subset of $L$ of labels of the productions permitted on the next step of derivation if $Q=F$("false"); it is called a *permissible set in case of failure*.

A finite set of strings from $V_T^*$ and formulas from $\pi_n$ , in which each formal parameter (for every entry of a terminal symbol into a string) is attributed with a value from $E$ and each symbol $f$ from $Fvar$ is matched with a mapping $h(f)$, serves as a *derivation result*.

Derivation in controlled grammar takes place as follows. A symbol $S$ serves as the start of derivation, where its formal parameters are provided with initial mappings $h(f)$ are specified for all $f$ from $Fvar$. In the role of the *initial permissible set* of productions we take the entire set $L$. To a current string we apply each of the productions of the current permissible set, the symbol $A$ for which enters into the string. As a result of applying a production, a new string and a new permissible set are formed. Later on derivation for each of the strings obtained from a given one takes place independently.

If none of the productions from permissible set can be applied, then derivation of the given string is discontinued. If this string consists only of terminal symbols, then it goes into the set of *derivation results*, otherwise it is discarded.

The application of a production takes place as follows. We choose the leftmost entry of the symbol $A$ in the string. We compute the value of the predicate $Q$. If $Q=F$, the $F_F$

becomes the permissible set, and the application of the production is ended. If $Q=T$, then the symbol $A$ is replaced by the string $B$; we carry out computation of the values of all formulas from $\pi_k$ corresponding to the parameters of the symbols, and the parameters assume new values thus computed. New mappings $h(f)$ ($f$ from $Fvar$) are specified by means of formulas from $\pi_n$; the permissible set is furnished by $F_T$, and application of the production is ended. (In the record of the production the formulas from $\pi_n$ leaving $h(f)$ unaltered are omitted.)

In constructions with which the controlled grammar is provided, it is easy to observe analogies with the programming language SNOBOL-4.

A **language L[G] generated by the controlled grammar G** is the union of all the sets which are the derivation results in this grammar.

## 7. ONE-DIMENSIONAL LINGUISTIC GEOMETRY

Here, we define the lowest level language of the hierarchy of languages. It serves as a building block to create the upper-level languages [26, 27]. The lowest level language actually formalizes a notion of a path between two points for the certain element of the System. An element might follow this path to achieve the goal connected with the ending point.

A *trajectory* for an element p of P with the beginning at x of X and the end at the y of X ($x \neq y$) with a length $l$ is a following string of symbols with parameters, points of X:
$$t_0 = a(x)a(x_1)...a(x_l),$$
where each successive point $x_{i+1}$ is reachable from the previous point $x_i$: $R_p(x_i, x_{i+1})$ holds for $i = 0,1,..., l-1$; element p stands at the point x: ON(p)=x. We denote $t_p(x, y, l)$ the set of trajectories in which p, x, y, and $l$ coincide. $P(t_0)=\{x, x_1, ..., x_l\}$ is the set of parametric values of the trajectory $t_0$.

Fig. 2. An interpretation of shortest and admissible trajectories.
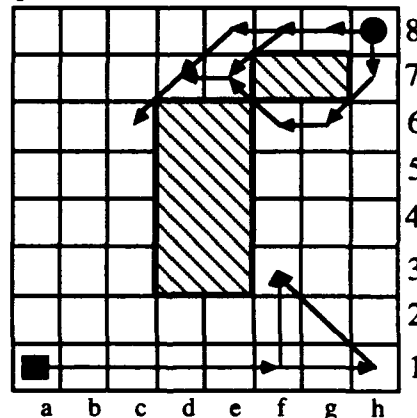


In order to illustrate this definition we consider the example from the robot control model (Fig. 2). Here the set of X (from Section 4) corresponds to the set of squares with coordinates: a1, b1, c1, ..., h8 excluding squares in the shaded area, which represent a restricted district. WFF ON(p)=x designate squares x were robots p stand in given state. Relations $R_p(x, y)$ designate moving capabilities of different robots, i.e., this relation holds for all squares y which are reachable from x in one step. For example for the robot C on h8 squares g8 and h7 are reachable; it has moving capabilities similar to King from the game of chess. The second robot S can move like Queen from the chess. Three trajectories of the

robot C leading from point h8 to c6 are shown in Fig. 2. Robot S has two trajectories leading from a1 to f3. Arrows mark points (squares) where robots have to stop during the motion along these trajectories. These points correspond to the values of parametric symbols for each trajectory.

A *shortest trajectory* t of $t_p(x, y, l)$ is the trajectory of minimum length for the given beginning x, end y and element p.

In Fig. 2, the two trajectories of robot C, $a(h8)a(g8)a(f8)a(e8)a(d7)a(c6)$ and $a(h8)a(g8)a(f8)a(e7)a(d7)a(c6)$, are the shortest trajectories. All the trajectories of the robot S shown in Fig. 2 are the shortest trajectories: $a(a1)a(f1)a(f3)$ and $a(a1)a(h1)a(f3)$. Reasoning informally, an analogy can be set up: the shortest trajectory is an analogous to a straight line segment connecting two points in a plane. Let us consider an analogy to a k-element segmented line connecting these points.

An *admissible trajectory of degree k* is the trajectory which can be divided into k shortest trajectories; more precisely there exists a subset $\{x_{i_1}, x_{i_2}, ..., x_{i_{k-1}}\}$ of $P(t_0)$, $i_1 < i_2 < ... < i_{k-1}$, $k \le l$, such that corresponding substrings $a(x_0)...a(x_{i_1})$, $a(x_{i_1})...a(x_{i_2})$, ..., $a(x_{i_{k-1}})...a(x_l)$ are the shortest trajectories.

Shortest and admissible trajectories of degree 2 play a special role in many problems. Obviously, every shortest trajectory is an admissible trajectory at the same time, but of course, converse statement is not true. There exist admissible trajectories, e.g., of degree 2, which are not shortest. An example of such a trajectory $a(h8)a(h7)a(g6)a(f6)a(e7)a(d7)a(c6)$ is shown in Fig. 2. As a rule, elements of the System should move along the shortest paths. In case of an obstacle, the element should move around this obstacle by tracing some intermediate point aside (e.g. point h7 in Fig. 2) and going to and from this point to the end along the shortest trajectories. Thus, in this case, an element should move along the admissible trajectory of degree 2.

A *Language of Trajectories* $L_t^H(S)$ for the Complex System in a state S is the set of all the shortest and admissible (degree 2) trajectories of the length less or equal H. This language also includes the empty trajectory e (of the length 0).

Properties of the Complex System permit to define (in general form) and study formal grammars for generating the Language of Trajectories as a whole along with its subsets: shortest and admissible (degree 2) trajectories.

## 8. GENERATION OF TRAJECTORIES

Consider the following controlled grammar for the Complex System with symmetric relation of reachability $R_p$ (Table 1):

**Table 1. A Grammar of Shortest Trajectories $G_t^{(1)}$**

| L | Q | Kernel, $\pi_k$ | $\pi_n$ | $F_T$ | $F_F$ |
|---|---|---|---|---|---|
| 1 | $Q_1$ | $S(x,y,l) \to A(x,y,l)$ | | two | $\emptyset$ |
| $2_i$ | $Q_2$ | $A(x,y,l) \to a(x)A(next_i(x,l), y, f(l))$ | | two | 3 |
| 3 | $Q_3$ | $A(x,y,l) \to a(y)$ | | $\emptyset$ | $\emptyset$ |

$V_T = \{a\}$

$V_N = \{S, A\}$

$V_{PR}$: $Pred = \{Q_1, Q_2, Q_3\}$,

$$Q_1(x, y, l) = (MAP_{x,p}(y)=l) \quad (0 < l < n)$$
$$Q_2(l) = (l \geq 1)$$
$$Q_3 = T$$

$Var = \{x, y, l\}$

$F = Fcon \cup Fvar$,

$\qquad Fcon=\{f, next_1,...,next_n\} \quad (n=|X|)$,

$\qquad\qquad f(l)=l-1, \quad D(f)=Z_+\backslash\{0\}$

$\qquad\qquad (next_i$ is defined below)

$\qquad Fvar=\{x_0,y_0,l_0,p\}$

$E = Z_+ \cup X \cup P$

$Parm$: $S \rightarrow Var$, $A \rightarrow Var$, $a \rightarrow \{x\}$

$L = \{1,3\} \cup two$, $two=\{2_1,2_2,...,2_n\}$

**At the beginning** of derivation:

$\qquad x=x_0, y=y_0, l=l_0, x_0$ from X, $y_0$ from X, $l_0$ from $Z_+$, p from P.

Function $next_i$ is defined as follows:

$D(next_i)= X \times Z_+ \times X^2 \times Z_+ \times P$

$SUM=\{v \mid v$ from X, $MAP_{x_0,p}(v)+MAP_{y_0,p}(v)=l_0\}$,

$ST_k(x)=\{v \mid v$ from X, $MAP_{x,p}(v)=k\}$,

$MOVE_f(x)$ is an intersection of the following sets:

$\qquad S\Gamma_1(x)$, $ST_{l_0-l+1}(x_0)$ and SUM.

If $MOVE_f(x)=\{m_1, m_2, ...,m_r\}\neq \emptyset$ then

$\quad next_i(x, l)=m_i$ for $i\leq r$ and

$\quad next_i(x, l)=m_r$ for $r<i\leq n$,

**otherwise**

$\quad next_i(x, l)=x$.

Fig. 3. An interpretation of the algorithm for $next_i$ for the grammar $G_t^{(1)}$.
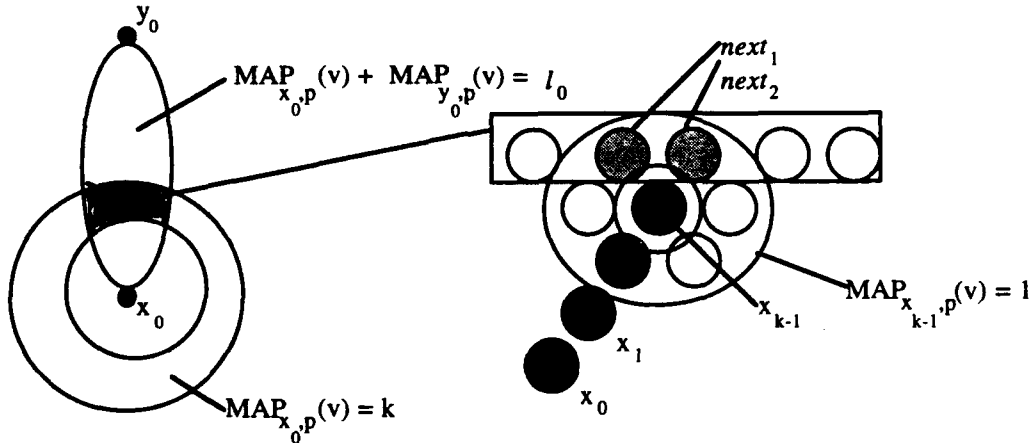


**THEOREM.** The shortest trajectories from point x to point y of the length $l_0$ for the element p on x (i.e., ON(p)=x) exist if and only if the distance of these points is equal $l_0$:

$$MAP_{x_0,p}(y_0)=l_0, \qquad\qquad (8.1)$$

where $l_0<2n$, n is the number of points in X. If the relation $R_p$ is symmetric, i.e., for all x from X, y from X and p from P $R_p(x, y)=R_p(y, x)$, then all the shortest trajectories $t_p(x_0, y_0, l_0)$ can be generated by the grammar $G_t^{(1)}$ (Table 1, Fig. 3).

***Proof.*** We assume that $t_0$ from $t_p(x_0, y_0, l_0)$ exists and is shortest. We shall prove (8.1). The proof is carried out by induction with respect to $l_0$.

In the case of $l_0=1$ the statement is easily verified.

We assume that for $l_0 < m$ the statement is true.

Let $l_0=m$ and $t_m$ from $t_p(x_0, y_0, m)$ be the shortest. We shall prove that $MAP_{x_0,p}(y_0)=m$. Let's consider the *shortened* trajectory $t_{m-1}$ from $t_p(x_0, x_{m-1}, m-1)$, $t_{m-1}=a(x_0)a(x_1)...a(x_{m-1})$, which is obtained from $t_m$ after discarding the last symbol. If $t_m$ from $t_p(x_0, x_m, m)$ is the shortest ($x_m=y_0$), then $t_{m-1}$ is also shortest. But from the assumption it follows that $MAP_{x_0,p}(x_{m-1})=m-1$. From definition of MAP (see Section 5) it follows that $x_{m-1}$ belongs to

$M^{m-1}_{x_0,p}$. Since $R_p(x_{m-1}, y_0)$ is true, $y_0$ belongs to $(\bigcup_{j=1}^{m-1} M^j_{x_0,p}) \cup M^m_{x_0,p}$. If $y_0$ is from $\bigcup_{j=1}^{m-1} M^j_{x_0,p}$, then the trajectory $t_m$ is not the shortest one, since there exists a trajectory $t'$

from $t_p(x_0, y_0, j)$ of length $j \le m-1$. We have a contradiction. Thus, $y_0$ belongs to $M^m_{x_0,p}$, i.e., $MAP_{x_0,p}(y_0)=m$.

Conversely, let (8.1) *be true*. Let's show that *there exists a trajectory belonging to* $t_p(x_0, y_0, l_0)$, and that *it is the shortest trajectory.*

The proof will be carried out by induction. For $l_0=1$ the statement is obvious. Let it be true for $l_0 < m$.

Let now $l_0=m$ and $MAP_{x_0,p}(y_0)=m$. The shortest trajectory if exists can not be shorter then $m$, otherwise there exists $k_0 < m$ such that $MAP_{x_0,p}(y_0)=k_0$ (from the direct statement proved above), and we have a contradiction.

Let us construct the shortest trajectory belonging to $t_p(x_0, y_0, m)$. By definition of MAP there exists $x_{m-1}$ from

$M^{m-1}_{x_0,p}$ such that $R_p(x_{m-1}, y_0)=T$. But from the fact that $x_{m-1}$ belongs to $M^{m-1}_{x_0,p}$, we have $MAP_{x_0,p}(x_{m-1})=m-1$. Consequently, according to the induction hypothesis, there exists the shortest trajectory $a(x_0)a(x_1)...a(x_{m-1})$ of length $m-1$. In such a case the trajectory $a(x_0)a(x_1)...a(x_{m-1})a(y_0)$ of length $m$ will also be the shortest one.

*To complete the proof* of the theorem it remains for us to show that *all trajectories* $t_p(x_0, y_0, l_0)$ *are generated by the grammar* $G_t^{(1)}$ from Table 1, if $R_p$ is symmetric. This grammar, in accordance with definition of controlled grammars (Section 6), belongs to the class of controlled grammars. Note that the set of functional symbols *Fvar* in it is a set of four zero-arity functions p, $x_0$, $y_0$, $l_0$, i.e., $G_t^{(1)}=G(p, x_0, y_0, l_0)$. It is obvious that each of the strings generated by $G_t^{(1)}$ is a trajectory from $t_p(x_0, y_0, l_0)$. Indeed, for each string $a(x_0)a(x_1)...a(y_0)$ thus generated, the elements $x_i$ belong to $ST_i(x_0)=M^i_{x_0,p}$ (see Fig. 3), consequently, this string is the shortest trajectory.

To prove that all the shortest trajectories are generated by $G_t^{(1)}$ let us conduct the following preliminary discussion. As was already mentioned above, all substrings of the shortest trajectory are the shortest trajectories with the beginning at $x_0$ and ending at $x_i$ ($i=1, 2,...., l_0$). Taking into account the symmetry of the relation $R_p$, all *reversed* substrings with the beginning at $y_0$ and ending at $x_i$ ($i =l_0-1, l_0-2,...,1, 0$) will also be the shortest trajectories. Consequently, $x_i$ belongs to

$M^{l_0-i}_{y_0,p}$. This means that for any shortest trajectory $a(x_0)a(x_1)...a(y_0)$ from $t_p(x_0,y_0,l_0)$

$x_i$ belongs to the intersection of $M_{x_o,p}^i$ and $M_{y_o,p}^{l_o-i}$, i.e., $MAP_{x_o,p}(x_i)=i$ and $MAP_{y_o,p}(x_i)=l_o-i$, and, consequently,

$$MAP_{x_o,p}(x_i)+MAP_{y_o,p}(x_i)=l_o. \qquad (8.2)$$

Conversely, if for a certain x from X (8.2) takes place, then x necessarily enters into the set $\boldsymbol{P}(t_i)$ parametric values of at least one shortest trajectory $t_i$ from $t_p(x_o, y_o, l_o)$. This follows from the fact that $MAP_{x_o,p}(x) \geq 0$ and $MAP_{y_o,p}(x) \geq 0$, while their sum is equal to $l_o$. That is to say, there exists j $(0 \leq j \leq l_o)$, such that $MAP_{x_o,p}(x)=j$, $MAP_{y_o,p}(x)=l_o-j$. Then there exist two shortest trajectories $t^1$ from $t_p(x_o, x, j)$ and $t^2$ from $t_p(y_o, x, l_o-j)$. The trajectory $t^3$ from $t_p(x, y_o, l_o-j)$ constructed of the same symbols as $t_2$, but in the reversed order, will also be the shortest trajectory. The concatenation of $t^1$ and $t^2$ gives the sought shortest trajectory containing x.

Thus, any element of the set X enters into the set of parametric values $\bigcup_{t^i} \boldsymbol{P}(t^i)$

for all the shortest trajectories $t_i$ from $t_p(x_o, y_o, l_o)$ if and only if (8.2) is true. These arguments lay a basis for the algorithm for calculating the function $next_i(x, l)$ (Fig. 3).

Next we shall use induction again. Obviously, the grammar of trajectories generates the first symbol $a(x_o)$ of all shortest trajectories from $t_p(x_o, y_o, l_o)$. Assume that it generates the m first symbols of any shortest trajectory from $t_p(x_o, y_o, l_o)$. We shall show that it generates also the (m+1)st symbol $a(x_m)$.

We have: $MOVE(x_{m-1})$ is an intersection of $ST_1(x_{m-1})$, $ST_m(x_o)$ and SUM. Since $t_p(x_o, y_o, l_o)$ are the shortest trajectories, $x_m$ belongs to $ST_m(x_o)=M^m_{x_o,p}$. But $x_m$ also belongs to SUM, because of (8.2), and $x_m$ belongs to $ST_1(x_{m-1})=M^1_{x_{m-1},p}$ since $R_p(x_{m-1}, x_m)=T$ by definition of trajectory. Thus, $x_m$ belongs to $MOVE(x_{m-1})$, i.e., the (m+1)st symbol is generated by the grammar $G_t^{(1)}$.

The theorem is proved.

# 9. DISCUSSION OF RESULTS

This paper reports the results on investigation of geometrical properties of complex systems. It explores properties of the first-level subsystems, paths of elements, the so-called trajectories. These results are considered as contribution to the One-Dimensional Linguistic Geometry.

The investigation resulted in definition of a distance function between two points of the system as a "length of the shortest path between these points". It is interesting that distances between the same two points are different for different elements of the system. It takes place because usually paths for different elements are different, i.e., moving capabilities of different robots as well as maintainability characteristics of different power units are different.

The distance measurement allowed us to build the general formal grammar generating all the shortest paths between two points for the given element of the system, the shortest trajectories. There was proved the theorem (Section 8) which gives necessary and sufficient conditions for existence of a path (trajectory) between two points (for the given element); if such path does exist the theorem shows the actual length of the shortest path and confirms that grammar $G_t^{(1)}$ generates all the shortest paths. Analogous results were

obtained in case of obstacles: visible and invisible. In this case the so-called "admissible trajectories of degree 2", i.e., constructed of two shortest ones, can be generated by the $G_t^{(2)}$ grammar to go around the obstacles [25, 26]. The application of the Linguistic Geometry to the game of chess, robot control, and maintenance scheduling allowed for efficient implementation of the Language of Trajectories in these models [26].

The same generating tools can be used to generate higher level subsystems, the networks of paths, i.e., the Language of Trajectory Networks [27]. Even the Language of Translations [27, 28] describing the process of search can be generated by a similar type of grammars. Consequently, the investigation of the control of the search for an optimal operation of the complex system can be reduced to the investigation of properties of the specific formal grammars.

## REFERENCES

1. M.R. Garey and D.S.Johnson, *Computers and Intractability: A Guide to the Theory of NP–Completeness* ,W.H.Freeman and Co., San Francisco, (1991).

2. H. Simon, *The Sciences of the Artificial* , The MIT Press, Cambridge, MA, (1980).

3. M.D. Mesarovich, D. Macko, Y. Takahara Y., *Theory of Hierarchical Multilevel Systems*, Academic Press, New York, (1970).

4. M.M. Botvinnik, *Computers in Chess: Solving Inexact Search Problems. Springer Series in Symbolic Computation* , Springer-Verlag, New York , (1984).

5. N. Chomsky, Formal Properties of Grammars, in *Handbook of Mathematical Psychology*, ed. R. Luce, R. Bush, E. Galanter., 2, 323-418, John Wiley & Sons, New York, (1963).

6. S. Ginsburg, *The Mathematical Theory of Context-Free Languages* , McGraw Hill, New York, (1966).

7. D.E. Knuth, Semantics of Context-Free Languages, Mathematical Systems Theory, 2-2, 127–146, (1968).

8. D.J. Rozenkrantz, Programmed Grammars and Classes of Formal Languages, *Journal of the ACM*, 16-1, 107–131, (1969).

9. K.S. Fu, *Syntactic methods in pattern recognition, Mathematics in Science and Engineering*, Vol. 112, ed. Richard Bellman, Academic Press, New York, (1974).

10. K.S. Fu, *Digital pattern recognition*, Springer-Verlag, New York, (1980).

11. R.N. Narasimhan, Syntax–Directed Interpretation of Classes of Pictures, *Comm. of the ACM*, 9, 166–173, (1966).

12. T. Pavlidis, Linear and Context-Free Graph Grammars, *Journal of the ACM*, 19,11–22, (1972).

13. A.C. Shaw, A Formal Picture Description Scheme as a Basis for Picture Processing System, *Information and Control*, 19, 9–52, (1969).

14. J. Feder, Plex languages, *Information Sciences*, 3, 225-241, (1971).

15. J.L. Pfaltz and A. Rosenfeld, WEB grammars, in *Proc. of the 1st International Joint Conf. Artificial Intelligence* ,Washington, D.C., 609–619, (May 1969).

16. N.G. Volchenkov, An Interpreter of Context-Free Controlled Parametric Programmed Grammars, in: *Cybernetics Problems. Intellectual Data Banks*, ed. L.T. Kuzin, The USSR Academy of Sciences, Moscow, 147–157, (1979), [in Russian].

17. R.E. Fikes and N.J. Nilsson, STRIPS: A New Approach to the Application of Theorem Proving in Problem Solving, *Artificial Intelligence*, 2, 189–208, (1971).

18. E.D. Sacerdoti, Planning in a Hierarchy of Abstraction Spaces, *Artificial Intelligence*, 5-1, 115-135, (1974).

19. J. McCarthy and P.J. Hayes, Some Philosophical Problems from the Standpoint of Artificial Intelligence, *Machine Intelligence*, 4, 463–502, (1969).

20. N. J. Nilsson, *Principles of Artificial Intelligence*, Tioga Pub., Palo Alto, CA, (1980).

21. B. Stilman, Hierarchy of Formal Grammars for Solving Search Problems, in: *Proceedings of the International Workshop, Artificial Intelligence. Results and Prospects*, Moscow, 63–72 (1985), [in Russian].

22. A.I. Reznitskiy and B.M. Stilman, Use of Method PIONEER in Automating the Planning of Power-Generating Equipment Maintenance, *Automatics and Remote Control*, 11, 147-153, (1983), [in Russian].

23. B. Stilman, A Syntactic Structure for Complex Systems, *Proc. of the Second Golden West International Conference on Artificial Intelligence*, Reno, NE, 269–274, (June 1992).

24. B. Stilman, A Syntactic Approach to Geometric Reasoning about Complex Systems, *Proc. of the Fifth International Symposium on Artificial Intelligence*, Cancun, Mexico, 115-124, (Dec. 1992).

25. B. Stilman, A Geometry of Hierarchical Systems: Generating Techniques, *Proc. of the Ninth Israeli Symposium on Artificial Intelligence and Computer Vision*, Ramat Gan, Israel, 95-109, (Dec. 1992).

26. B. Stilman, A Linguistic Approach to Geometric Reasoning, *International Journal: Computers and Mathematics with Applications* , (1993), (to appear).

27. B. Stilman, Network Languages for Complex Systems, *International Journal: Computers and Mathematics with Applications* , (1993), (to appear).

28. B. Stilman, Translations of Network Languages, *International Journal: Computers and Mathematics with Applications* , (1993), (to appear).

# AN APPLICATION OF ROUGH SETS IN KNOWLEDGE SYNTHESIS

S.K.M. Wong, Y.Y. Yao[1] and L.S. Wang

Department of Computer Science, University of Regina
Regina, Saskatchewan, Canada S4S 0A2

**Abstract**

In this paper, we consider the fundamental issues in knowledge verification and synthesis by focusing on a special type of rule-based systems, which consists of a set of deterministic and non-deterministic decision rules. A set of *sound* and *complete* inference axioms is suggested. Based on these axioms, an efficient algorithm is developed for computing the closure and testing the consistency of the input rules.

# 1   INTRODUCTION

Knowledge verification and synthesis are two important processes in the design and implementation of intelligent systems as it is often necessary to validate and consolidate the input knowledge [3, 6, 8, 9, 18].

Many approaches for verification (validation) of knowledge in the rule-based systems were proposed [7]. In fact, many systems have been developed to identify inconsistent, redundant or missing rules in a knowledge base [7, 8, 9]. However, research on knowledge verification tends to be fragmentary in nature and unclear in scope and methodology [7].

In this paper, by adopting an axiomatic approach we analyze some of the issues in knowledge verification and synthesis. Our approach is similar to the method used for analyzing functional dependencies in relational databases [5]. Our study will focus on the knowledge base consisting of deterministic and non-deterministic rules [10, 11, 12, 15]. Based on the notion of logical implication, we introduce a set of inference axioms for deriving new rules from the input rules. This process of

---

[1]Currently at Department of Mathematical Sciences, Lakehead University, Thunder Bay, Ontario, Canada P7B 5E1

synthesis is similar to inferring new functional dependencies in a relational database. The proposed set of axioms is related to that suggested by Bundy [1, 2] for incidence calculus. In particular, we show that our inference axioms are both *sound* and *complete*. Using these axioms, we also develop an algorithm to synthesize the input rules.

# 2  ROUGH SETS AND DECISION RULES

In this section, we extend the notion of rough sets based on a *compatibility* relation between the elements of two sets. The lower and upper approximations of a *concept* suggest two kinds of decision rules for uncertain reasoning.

## 2.1  ROUGH SETS INDUCED BY A COMPATIBILITY RELATION

Let $W = \{w_1, w_2, \ldots, w_m\}$ and $\Theta = \{\theta_1, \theta_2, \ldots, \theta_n\}$ denote two finite non-empty sets of interest. The set $W$ may be regarded as a frame for representing evidence [13], whose elements are *descriptions* or *situations* [15]. On the other hand, the set $\Theta$ may be interpreted as a frame for representing propositions, whose elements are elementary hypotheses. The relationship between the elements of these two frames can be described by a compatibility relation between $W$ and $\Theta$, a subset of the Cartesian product $W \times \Theta$. A description or situation $w \in W$ is said to be *compatible* with a hypothesis $\theta \in \Theta$, written $w \, \Re \, \theta$, if $w$ does not contradict $\theta$. Semantically speaking, compatibility is symmetric: $w$ is compatible with $\theta$ if and only if $\theta$ is compatible with $w$. Without lose of generality, we may assume that for any $w \in W$ there exists at least one $\theta \in \Theta$ such that $w \, \Re \, \theta$, and vice versa. For example, consider a diagnostic system, in which $W$ denotes a set of symptoms, and $\Theta$ a set of diseases. In this case, a symptom $w \in W$ is said to be compatible with a disease $\theta \in \Theta$, if a patient who suffers from the disease $\theta$ has symptom $w$. Another example can be found in incidence calculus [17], where a situation $w \in W$ is compatible with a hypothesis $\theta \in \Theta$ if $w$ does not rule out the possibility that $\theta$ is true.

Suppose we want to characterize a subset $A \subseteq \Theta$ in terms of the elements in $W$. Given a compatibility relation $\Re$, one can define a mapping $\Gamma_{\Re}$ which assigns a subset $\Gamma_{\Re}(w) \subseteq \Theta$ to every $w \in W$ as:

$$\Gamma_{\Re}(w) = \{\theta \in \Theta \mid w \, \Re \, \theta\}. \tag{1}$$

Conversely, for any subset $A \subseteq \Theta$, one can define the lower preimage $\underline{\Re}(A)$ and the upper preimage $\overline{\Re}(A)$ of $A$ as:

$$\underline{\Re}(A) \;=\; \{w \in W \mid \Gamma_{\Re}(w) \subseteq A\}\,, \tag{2}$$

$$\overline{\Re}(A) \;=\; \{w \in W \mid \Gamma_{\Re}(w) \cap A \neq \emptyset\}\,. \tag{3}$$

The set $\underline{\Re}(A)$ consists of *all* those $w$'s compatible with *only* the elements in $A$. The set $\overline{\Re}(A)$ contains all those $w$'s, each of which is compatible with at least *one* element in $A$. Obviously, $\underline{\Re}(A) \subseteq \overline{\Re}(A)$ for any $A \subseteq \Theta$. The pair $(\underline{\Re}(A), \overline{\Re}(A))$ can be viewed as a *rough set* of $A$ induced by the compatibility relation $\Re$ [10]. $\underline{\Re}(A)$ is referred to as the lower approximation (the greatest lower bound) of $A$, and $\overline{\Re}(A)$ is called the upper approximation (the smallest upper bound) of $A$.

It can be verified that for any subsets $A, B \subseteq \Theta$, the following properties hold [10, 13]:

(P1)  $\underline{\Re}(A \cap B) = \underline{\Re}(A) \cap \underline{\Re}(B)$,

(P2)  $\underline{\Re}(A \cup B) \supseteq \underline{\Re}(A) \cup \underline{\Re}(B)$,

(P3)  $\overline{\Re}(A \cap B) \subseteq \overline{\Re}(A) \cap \overline{\Re}(B)$,

(P4)  $\overline{\Re}(A \cup B) = \overline{\Re}(A) \cup \overline{\Re}(B)$,

(P5)  $\underline{\Re}(\neg A) = W - \overline{\Re}(A)$,  $\overline{\Re}(\neg A) = W - \underline{\Re}(A)$,

(P6)  $A \supseteq B \Longrightarrow (\underline{\Re}(A) \supseteq \underline{\Re}(B), \overline{\Re}(A) \supseteq \overline{\Re}(B))$,

(P7)  $\underline{\Re}(\Theta) = \overline{\Re}(\Theta) = W$,

(P8)  $\underline{\Re}(\emptyset) = \overline{\Re}(\emptyset) = \emptyset$.

Note that these properties are not independent. In fact, (P1), $\underline{\Re}(\Theta) = W$ and $\underline{\Re}(\emptyset) = \emptyset$ are independent and sufficient for describing the lower approximations. Likewise, (P4), $\overline{\Re}(\Theta) = W$ and $\overline{\Re}(\emptyset) = \emptyset$ are independent and sufficient for describing the upper approximations.

Instead of using equations (2) and (3), the lower and upper approximations can be equivalently defined by the following formulas:

$$\underline{\Re}(A) \;=\; \bigcup_{B \subseteq A} j_{\Re}(B), \tag{4}$$

$$\overline{\Re}(A) \;=\; \bigcup_{A \cap B \neq \emptyset} j_{\Re}(B), \tag{5}$$

where the mapping, $j_{\Re} : 2^{\Theta} \to 2^{W}$, is called the *basic set assignment* defined by

$$j_{\Re}(B) = \{w \mid \Gamma(w) = B\}. \tag{6}$$

The basic set assignment $j_\mathfrak{R}$ satisfies the following properties.

(B1)    $j_\mathfrak{R}(\emptyset) = \emptyset$,

(B2)    $\bigcup_{A \subseteq \Theta} j_\mathfrak{R}(A) = W$,

(B3)    for any $A, B \in 2^\Theta, A \neq B \implies j_\mathfrak{R}(A) \cap j_\mathfrak{R}(B) = \emptyset$.

## 2.2 DECISION RULES INDUCED BY ROUGH SETS

Based on the lower and upper approximations of a proposition $A \subseteq \Theta$, one can define two kinds of decision rules [12]. For every $w \in \underline{\mathfrak{R}}(A)$, $A$ contains *all* the $\theta$'s that are compatible with $w$. Thus, whenever $w \in \underline{\mathfrak{R}}(A)$, we can conclude that the proposition represented by $A$ is true, namely, the proposition $\{w\}$ implies $A$, written $\{w\} \rightarrow A$. That is, the lower preimage of $A$ defines a deterministic decision rule, "$\underline{\mathfrak{R}}(A)$ definitely implies $A$", written $\underline{\mathfrak{R}}(A) \rightarrow A$. On the other hand, whenever $w \in \overline{\mathfrak{R}}(A)$, proposition $A$ is possibly true, namely, $\{w\}$ possibly implies $A$, written $\{w\} \rightsquigarrow A$. This means that the upper preimage of $A$ defines a non-deterministic decision rule, "$\overline{\mathfrak{R}}(A)$ possibly implies $A$", written $\overline{\mathfrak{R}}(A) \rightsquigarrow A$.

When the compatibility relation is given, it is a straightforward task to construct the deterministic and non-deterministic decision rules as described above. Alternatively, one may use an inductive method to learn these rules from a number of examples which implicitly define the compatibility relationships between the elements of two frames [12]. The decision rules obtained by these methods are always consistent, and no synthesis is required. However, in many practical situations the decision rules are neither learned from the examples nor derived from a compatibility relation. Instead, the rules are given by the experts. Since these rules are specified *separately* for the individual propositions, inconsistency may occur. That is, there may exist contradictions among the input rules. Consider the following rules, for example, $r_1 : \{w_1, w_2\} \rightarrow \{\theta_1\}$ and $r_2 : \{w_1, w_3\} \rightsquigarrow \{\theta_1\}$. Obviously, these two rules contradict each other because rule $r_1$ says that if the description is $w_2$, $\theta_1$ is true, whereas rule $r_2$ says that if the description is $w_2$, $\theta_1$ is not true. Therefore, we need a method for testing the consistency of the rules provided by the experts.

It is also important to note that new rules can be logically inferred from a given set of rules. For instance, from $\{w_1\} \rightarrow \{\theta_1, \theta_2\}$, we know that if the description is $w_1$, either $\theta_1$ or $\theta_2$ is true. On the other hand, from $\{w_1\} \rightarrow \{\theta_1, \theta_3\}$, we can conclude that if the description is $w_1$, either $\theta_1$ or $\theta_3$ is true. These two rules implicitly imply

that $\{w_1\} \rightarrow \{\theta_1\}$ holds. Thus, an inference mechanism for synthesizing the input rules are required.

# 3 KNOWLEDGE SYNTHESIS

Before presenting a method for the verification and synthesis of input rules, we first define the notions of logical implication and consistency in our approach.

## 3.1 LOGICAL IMPLICATION AND CONSISTENCY

Let $\underline{F}(A)$ and $\overline{F}(A)$ denote subsets of $W$. A set of decision rules $F = \{\underline{F}(A) \rightarrow A, \overline{F}(A) \rightsquigarrow A | A \in 2^\Theta\}$ given by the experts can be viewed as a pair of mappings, $\underline{F}$ and $\overline{F}$, from $2^\Theta$ to $2^W$. Without loss of generality, we may assume that $\underline{F}(\Theta) = W$ holds. Also, if there is no information about proposition $A$, we assume that $\underline{F}(A) = \emptyset$ and $\overline{F}(A) = W$. We call $F$ an *assignment*. In $F$, each $\underline{F}(A) \rightarrow A$ represents a deterministic rule, and each $\overline{F}(A) \rightsquigarrow A$ represents a non-deterministic rule. The deterministic rule, $\underline{F}(A) \rightarrow A$, indicates that for every $w \in \underline{F}(A)$, $A$ contains *all* the $\theta$'s that are compatible with $w$. However, $\underline{F}(A)$ does not necessarily contain *all* the $w$'s that are compatible with only the $\theta$'s in $A$. The non-deterministic decision rule, $\overline{F}(A) \rightsquigarrow A$, says that *only* those descriptions in $\overline{F}(A)$ may imply $A$. That is, whenever $w \notin \overline{F}(A)$, $w$ is not compatible with any $\theta$ in $A$. However, there may exist some $w$'s in $\overline{F}(A)$ not compatible with any $\theta$ in $A$.

Let $\Re$ denote the *true* compatibility relation that defines the relationships between the individual elements of $\Theta$ and $W$. Given an assignment $F$, suppose $\underline{F}(A) \subseteq \overline{F}(A)$ for all $A \in 2^\Theta$. Then, by the definitions of $\underline{\Re}(A)$ and $\overline{\Re}(A)$, $\underline{F}(A) \subseteq \underline{\Re}(A) \subseteq \overline{\Re}(A) \subseteq \overline{F}(A)$ holds for all $A \in 2^\Theta$. Also, there may exist a number of compatibility relations $\Re_i$'s satisfying the condition: $\underline{F}(A) \subseteq \underline{\Re_i}(A) \subseteq \overline{\Re_i}(A) \subseteq \overline{F}(A)$ for all $A \in 2^\Theta$. Clearly, any of these $\Re_i$'s could be the true compatibility relation.

For convenience, we will use $X$, $Y$, $Z$ to denote subsets of $W$ and $A$, $B$, $C$ to denote subsets of $\Theta$ in subsequent discussions.

**Definition 1.** A compatibility relation $\Re$ *satisfies* a deterministic decision rule, $X \rightarrow A$, if $X \subseteq \underline{\Re}(A)$; $\Re$ *satisfies* a non-deterministic decision rule, $X \rightsquigarrow A$, if $\overline{\Re}(A) \subseteq X$. A compatibility relation $\Re$ satisfies an assignment $F$, if $\Re$ satisfies every decision rule in $F$.

**Definition 2.** An assignment $F$ *logically* implies a deterministic decision rule, $X \rightarrow A$, written $F \models \{X \rightarrow A\}$, if for every compatibility relation $\Re$ satisfying

$F$, $\Re$ also satisfies $X \to A$. Similarly, $F$ logically implies a non-deterministic rule $X \leadsto A$, written $F \models \{X \leadsto A\}$, if for every compatibility relation $\Re$ satisfying $F$, $\Re$ also satisfies $X \leadsto A$. We use $F^*$ to denote the set of all the decision rules that are logically implied by an assignment $F$.

**Example 1.** Let $\Theta = \{\theta_1, \theta_2, \theta_3\}$ and $W = \{w_1, w_2, w_3\}$. Consider an assignment $F$ given below:

| $A$ | $\underline{F}(A)$ | $\overline{F}(A)$ |
|---|---|---|
| $\emptyset$ | $\emptyset$ | $\emptyset$ |
| $\{\theta_1\}$ | $\{w_1\}$ | $\{w_1\}$ |
| $\{\theta_2\}$ | $\{w_2\}$ | $\{w_2,\ w_3\}$ |
| $\{\theta_3\}$ | $\emptyset$ | $\{w_3\}$ |
| $\{\theta_1, \theta_2\}$ | $\{w_2\}$ | $W$ |
| $\{\theta_1, \theta_3\}$ | $\{w_1\}$ | $W$ |
| $\{\theta_2, \theta_3\}$ | $\{w_2,\ w_3\}$ | $\{w_2,\ w_3\}$ |
| $\{\theta_1, \theta_2, \theta_3\}$ | $W$ | $W$ |

There are only two compatibility relations $\Re_1$ and $\Re_2$ satisfying the above assignment, namely:

$$\Re_1 : \quad w_1\ \Re_1\ \theta_1,\ w_2\ \Re_1\ \theta_2,\ w_3\ \Re_1\ \theta_3;$$

$$\Re_2 : \quad w_1\ \Re_2\ \theta_1,\ w_2\ \Re_2\ \theta_2,\ w_3\ \Re_2\ \theta_3,\ w_3\ \Re_2\ \theta_2.$$

Note that, both $\Re_1$ and $\Re_2$ satisfy the decision rules, $\{w_1,\ w_2\} \to \{\theta_1,\ \theta_2\}$ and $\{w_1,\ w_3\} \leadsto \{\theta_1,\ \theta_3\}$. By definition, these two rules are therefore logically implied by $F$.

**Definition 3.** Let $F$ be an assignment. If there exists a compatibility relation satisfying $F$, then $F$ is *consistent*; otherwise $F$ is *inconsistent*.

**Example 2** Let $\Theta = \{\theta_1, \theta_2, \theta_3\}$ and $W = \{w_1, w_2, w_3\}$. Suppose the assignment $F$ is defined by:

$$\underline{F}(\{\theta_1\}) = \overline{F}(\{\theta_1\}) = \{w_1\},$$
$$\underline{F}(\{\theta_2\} = \overline{F}(\{\theta_2\}) = \{w_2\},$$
$$\underline{F}(\{\theta_1, \theta_2\}) = \{w_1, w_3\},$$
$$\overline{F}(\{\theta_1, \theta_2\}) = \{w_1, w_3\}.$$

From $\underline{F}(\{\theta_2\}) = \{w_2\}$, we obtain $w_2 \Re \theta_2$. On the other hand, $\overline{F}(\{\theta_1, \theta_2\}) = \{w_1, w_3\}$ implies that $\neg(w_2 \Re \theta_1)$ and $\neg(w_2 \Re \theta_2)$. This means that no compatibility relation would satisfy this assignment. That is, this assignment is inconsistent.

## 3.2   COMPUTATION OF THE CLOSURE

Let $F^*$ denote the set of all decision rules that can be logically implied by a given assignment $F$. Our objective is compute $F^*$ by using a set of inference axioms. Our approach is similar to that for finding the closure of a set of functional dependencies in a relational database [5].

For our purpose, the inference axioms can be expressed as:

$(I_1)$     $X \rightsquigarrow A$ and $Y \rightarrow \neg A \Longrightarrow X - Y \rightsquigarrow A$.

$(I_2)$     $X \rightsquigarrow \neg A$ and $Y \rightarrow A \Longrightarrow Y \cup (W - X) \rightarrow A$.

$(I_3)$     $X \rightsquigarrow A, Y \rightsquigarrow B$ and $Z \rightsquigarrow A \cap B \Longrightarrow X \cap Y \cap Z \rightsquigarrow A \cap B$.

$(I_4)$     $X \rightarrow A, Y \rightarrow B$ and $Z \rightarrow A \cap B \Longrightarrow (X \cap Y) \cup Z \rightarrow A \cap B$.

$(I_5)$     $X \rightarrow A \cap B$ and $Y \rightarrow A \Longrightarrow X \cup Y \rightarrow A$.

$(I_6)$     $X \rightarrow A \Longrightarrow Y \rightarrow A$ for any $Y \subseteq X$.

$(I_7)$     $X \rightsquigarrow A \Longrightarrow Y \rightsquigarrow A$ for any $Y \supseteq X$.

Although these axioms $(I_1)$-$(I_5)$ are similar to those introduced by [1], we express them here as inference rules.

**Definition 4.** Let $I$ denote a set of inference axioms. With respect to $I$, the *closure* of an assignment $F$, written $F_I^+$, is the *smallest* set containing $F$ such that the no axiom cannot be applied to the set to yield an decision rule not in the set.

**Definition 5.** We say that a set of inference axioms $I$ is *sound* if any decision rule $X \rightarrow A$ or $X \rightsquigarrow A$ in $F_I^+$ is in $F^*$, i.e., $F_I^+ \subseteq F^*$. We say that $I$ is *complete* if $F^* \subseteq F_I^+$.

It can be proved that the above set of axioms $I^0 = \{I_1, I_2, I_3, I_4, I_5, I_6, I_7\}$ is both sound and complete [16].

Let $F_{I^0}^+$ denote the closure of $F$ with respect to $I^0$. For every $A$ in $2^\Theta$, there may exist many deterministic and non-deterministic decision rules in $F_{I^0}^+$ with $A$ at the right-hand side. Based on $(I_4)$, we know that if $X_1 \rightarrow A$ and $X_2 \rightarrow A$ are in $F_{I^0}^+$, $X_1 \cup X_2 \rightarrow A$ is in $F_{I^0}^+$. Similarly, $(I_3)$ implies that if $Y_1 \rightsquigarrow A$ and $Y_2 \rightsquigarrow A$ are in $F_{I^0}^+$, $Y_1 \cap Y_2 \rightsquigarrow A$ is in $F_{I^0}^+$. Therefore, for any $A$ in $2^\Theta$, there exist a deterministic

decision rule $\inf(A) \to A$ and a non-deterministic decision rule $\sup(A) \rightsquigarrow A$ in $F_{j^0}^+$ such that whenever $X \to A$ is in $F_{j^0}^+$, $X \subseteq \inf(A)$, and whenever $X \rightsquigarrow A$ is in $F_{j^0}^+$, $\sup(A) \subseteq X$.

**Definition 6.** For any $A$ in $2^\Theta$, $\inf(A) \to A$ is called the *max* deterministic decision rule of $A$, if for any $X \to A$ in $F_{j^0}^+$, $X \subseteq \inf(A)$; $\sup(A) \rightsquigarrow A$ is called the *min* non-deterministic decision rule of $A$, if for any $X \rightsquigarrow A$ in $F_{j^0}^+$, $\sup(A) \subseteq X$. The set of all the *max* deterministic and *min* non-deterministic decision rules, $F_0 = \{\inf(A) \to A, \sup(A) \rightsquigarrow A | A \in 2^\Theta\}$, is called the *max-min* assignment of $F$.

Based on the notion of max-min assignments, there is a simple way for checking the consistency of an assignment. An assignment is inconsistent if and only if $\underline{F_0}(A) \not\subseteq \overline{F_0}(A)$ for some $A$ in $2^\Theta$ [16].

## 3.3 CONSTRUCTION OF THE MAX-MIN COVER

The process of synthesis is to derive a new set of decision rules which have desir_.ole properties and *cover* the original set of rules.

**Definition 7.** Consider *two sets* of decision rules (assignments) $G$ and $F$. $G$ is *equivalent* to $F$ if $G^* = F^*$. $G$ is a *cover* of $F$ with respect to a s und and complete set $I$ of inference axioms, if $G_I^+ = F^*$.

Based on the above definition, the max-min assignment $F_0$ is obviously a cover of the original assignment $F$. We call $F_0$ a max-min cover. Moreover, such a cover satisfies the properties (P1)-(P8), if one replaces $\underline{\Re}$ by $\underline{F_0}$, and $\overline{\Re}$ by $\overline{F_0}$. Recall that one can equivalent define the lower and upper approximations in terms of the basic set assignment. This suggests that it may be easier to compute the basic set assignment $j_{F_0}$ than to compute $F_0$ directly from the inference axioms. Given below is an algorithm for constructing the max-min cover.

**Input:** $F = \{\underline{F}(A) \to A,\ \overline{F}(A) \rightsquigarrow A\ |$
$\qquad A \in 2^\Theta,\ \underline{F}(A) \neq \emptyset$ and $\overline{F}(A) \neq W\}$;
1. **for** each rule $\overline{F}(A) \rightsquigarrow A$ in $F$ **do**
$\qquad \underline{F}'(\neg A) = \underline{F}(\neg A) \cup (W - \overline{F}(A))$;
2. **for** each $w_k \in W$ **do**
$\qquad$ Find all the $A$'s where $\underline{F}'(A) \neq \emptyset$ such that
$\qquad w_k \in \underline{F}'(A)$, say, $A_1, A_2, ..., A_l$;

148

if $\quad A_1 \cap A_2 \cap \ldots \cap A_l = \emptyset \quad$ **then**

  exits to *inconsistent*;

**else**

  $j(A_1 \cap A_2 \cap \ldots \cap A_l) = j(A_1 \cap A_2 \cap \ldots \cap A_l) \cup \{w_k\};$

  (Initially, $j(A_1 \cap A_2 \cap \ldots \cap A_l) = \emptyset$.)

3. **Output:** $j$.

Note that if $\underline{F}(\neg A)$ is not assigned a value in the input, we may assume $\underline{F}(A) = \emptyset$. This procedure exits to *inconsistent* if and only if the input assignment $F$ is inconsistent; otherwise it outputs the basic set assignment of the max-min cover. The desired decision rules can be easily constructed from formulas (4) and (5).

# 4 CONCLUSION

In this paper, we have taken an axiomatic approach to investigate the fundamental issues in knowledge verification and synthesis. Our approach shares many salient features of the methods for analyzing functional dependencies in a relational database. A set of sound and complete inference axioms has been suggested. Based on these axioms, an efficient algorithm has been developed for computing the closure and testing the consistency of the input decision rules.

Although our discussion has focused on a special type of deterministic and non-deterministic decision rules, the proposed method can be applied to other rule-based systems.

# REFERENCES

[1] Bundy, A. (1985). Incidence calculus: a mechanism for probabilistic reasoning. *Journal of Automated Reasoning,* **1**, 263-283.

[2] Bundy, A. (1986). Correctness criteria of some algorithms for uncertain reasoning using incidence calculus *Journal of Automated Reasoning,* **2**, 109-126.

[3] Hayes-roth, F. (1985). Rule-based systems. *Communications of the ACM,* **28**, 921-932.

[4] Lingras, P.J. and Wong, S.K.M. (1990). Two perspectives of the Dempster-Shafer theory of belief functions. *International Journal of Man-machine Studies*, **33**, 467-487.

[5] Maier, D. (1983). *The Theory of Relational Databases*, Rockville, Maryland: Computer Science Press.

[6] Marek, W. (1986). Completeness and consistency in knowledge base systems. *Proceedings of First International Conference on Expert Database Systems*, 75-82.

[7] Nazareth, D.L. (1989). Issues in the verification of knowledge in rule-based systems. *International Journal of Man-machine Studies*, **30**, 255-271.

[8] Nguyen, T.A., Perkins, W.A., Laffrey, T.J., and Pecora, D. (1985). Checking an expert systems knowledge base for consistency and completeness. *Proceedings of Ninth IJCAI*, 375-378.

[9] Nguyen, T.A., Perkins, W.A., Laffrey, T.J., and Pecora, D. (1987). Knowledge base verification. *AI Magazine*, **8**, 69-75.

[10] Pawlak, Z. (1982). Rough sets. *International Journal of Computer and Information and Sciences*, **11**, 341-356.

[11] Pawlak, Z. (1984). Rough classification. *International Journal of Man-Machine Studies*, **20**, 469-483.

[12] Pawlak, Z., Wong, S.K.M. and Ziarko, W. (1988). Rough sets: probabilistic versus deterministic approach. *International Journal of Man-Machine Studies*, **29**, 81-95.

[13] Shafer, G. (1976). *A Mathematical Theory of Evidence*, Princeton: Princeton University Press.

[14] Shafer, G. (1986). Belief functions and possibility measures. in *Analysis of Fuzzy Information*, Vol. I, J.C. Bezdek, Ed., CRC Press, 51-84.

[15] Wong, S.K.M., Ziarko, W. and Ye, R.L. (1986). Comparison of rough-set and statistical methods in inductive learning. *International Journal of Man-Machine Studies*, **24**, 53-72.

[16] Wong, S.K.M., Wang, L.S. and Yao, Y.Y. (1991). Knowledge synthesis in rule-based systems. Submitted from publication.

[17] Wong, S.K.M., Wang, L.S. and Yao, Y.Y. (1992). Interval structure: a framework for representing uncertain information. *Uncertainty in Artificial Intelligence: Proceedings of the 8th Conference*, 336-343

[18] Yager, R.R. and Larsen, H.L. (1991). On discovering potential inconsistencies in validating uncertain knowledge bases by reflecting on the input. *IEEE Transactions on System, Man, and Cybernetics*, **21**, 790-801.

# A RELATIONAL MODEL FOR IMPRECISE QUERIES[1]

Weining Zhang, Clement Yu,
Gaoming Wang, Tracy Pham and Hiroshi Nakajima[2]

## ABSTRACT

In this paper, we propose a fuzzy relational data model that enables a database system to answer imprecise queries often found in decision–making applications. The model is based on a fuzzy relation in which values of attributes are atomic and precise, while the membership of tuples may be fuzzy. A fuzzy tuple relational calculus and a fuzzy relational algebra, both provide new features, are defined and shown to be equivalent on their expressive power. The use of these query languages are illustrated by examples. Techniques that allow an implementation of the model on top of a standard relational databases system are discussed.

## 1   INTRODUCTION

The database support to decision–making applications in business, engineering, and science has become increasingly important. Such a support requires the database system to store and to process imprecise information that is inherent in human decision–making. The imprecision arises for several reasons. First, the natural language used in the decision–making process is itself imprecise. Second, the complexity of the real world and the lack of a complete knowledge of the data in the system make it too difficult, if not impossible, for a decision maker to describe precisely what he or she is interested in. Third, decisions are often made based on subjective and qualitative criteria which are inherently imprecise. Conventional database management systems assume precise data in databases and can only answer queries whose query conditions must be matched precisely by data in the answer. Therefore, they do not readily support decision–making applications.

Recently, the research on fuzzy databases combines the fuzzy sets theory, possibility theory, and fuzzy logic [18, 19]with the relational database technology to handle imprecise data and queries. Two approaches have been followed. The first approach [2, 3, 4, 5, 12, 14, 20] is to include in the database domain fuzzy values such as the null value, disjunctive values, and linguistic values represented by possibility distributions. Query languages including fuzzy relational algebra, calculus, and fuzzy SQL were proposed to model and to manipulate imprecise data. The resulting systems are integrated and the fuzziness of the data can be represented both at attribute level (that is, the domain of an attribute can contain fuzzy values) and at tuple level (that is, the membership of a tuple with respect to a relation can also be fuzzy). However, it usually requires the reconstruction of database management system, which is often too costly. Research followed this approach remains mainly theoretical. Issues of efficient implementation have not been adequately studied. For instance, in [12], although many theoretical aspects of a fuzzy database are discussed, the implementation is based on a fuzzy prolog language which is inherently tuple oriented and is not efficient in processing large volume of data. Recently, some attempt has been made [13] to implement a fuzzy database system on top of a

relational database system. However, the system currently provides only a programming interface to a fuzzy function library to perform limited fuzzy operations on data retrieved from the relational database. As such, the system represents a loosely coupled fuzzy data dictionary and a conventional relational database, and provides a less expressive query language as compared with the languages in this and other papers. The efficiency of query processing in this system is yet unknown.

The second approach [1, 6, 9, 10, 8, 15] is to extend current relational database management systems to support imprecise queries against precise data. The basic idea is to build front-end systems that allow users to express queries using imprecise conditions, involving fuzzy sets, while the data within the databases remain precise. Since such systems are based on the existing database management systems and their efficient implementations, they are easier to build and are more cost effective than the first approach. Some prototype systems following this approach are reported recently [1, 9, 8]. However, the extensions made by these systems are rather ad hoc. For example, In [8], a query system that supports fuzzy queries involving linguistic quantifiers and its implementation on top of dBase III Plus were described. The system was considered as adding additional commands to the dBase III Plus. But the data model based on which the system works is not defined formally. Especially, the linguistic quantifiers were not defined in terms of any formal language. The theory behind the implementation was unclear as well. We feel that a more systematic method to the problem is needed.

In this paper, we follow the second approach and propose a fuzzy relational data model that not only provides a theoretic foundation for imprecise (or fuzzy) query against precise data, but also allows implementation on top of existing relational database management systems. The data model is based on the fuzzy relation in which components of tuples have precise values, but each tuple belongs to the fuzzy relation to a certain degree. Two formal query languages, a fuzzy tuple relational calculus and a fuzzy relational algebra, are defined based on the fuzzy relation and provide stronger expressive power than those given in the literature. We also extend the notion of a safe calculus expression, which was not previously addressed in fuzzy databases literatures. Intuitively, a safe calculus expression denotes a finite relation whose tuples are constructed using symbols in the given fuzzy database and the given query. We prove that the the algebra and the calculus are equivalent, that is, any query expressible in one language is also expressible in the other. To our knowledge, such a result has not been previously reported[3]. The use of the query languages are illustrated through examples. Techniques that enable the implementation of the model on top of a standard relational database management system are also discussed.

The remainder of the paper is organized as follows. In Section 2, some background on fuzzy sets and imprecise queries is provided. In Section 3, the fuzzy relation is defined. In Section 4, we define a fuzzy tuple relational calculus. In Section 5, a fuzzy relational algebra is given. The theoretic results on the equivalence of the two formal query languages are given in Section 6. The discussions of the techniques for implementing a fuzzy relational database system on a standard relational DBMS is in Section 7. Section 8 concludes the paper.

## 2 FUZZY SETS AND IMPRECISE QUERIES

In this section, we briefly present concepts of fuzzy sets and imprecise queries.

### 2.1 FUZZY SETS

A *fuzzy set F* in domain $D$ is a collection of elements of $D$ such that each element $d \in D$ is associated with a degree $\mu_F(d)$ with which $d$ is a member of $F$. The degrees of the membership for the set $F$ is defined by a membership function $\mu_F : D \longrightarrow [0,1]$, which maps each element in the domain into a value in $[0,1]$, where 1 indicates a complete membership, 0 indicates a complete non-membership,

---

[3]Although in [12], it was mentioned that the equivalence of their fuzzy relational algebraic and calculus languages can be proved, neither a proof nor any reference on the issue was provided.

and other values indicate partial membership. For example, let the domain be the age spanning from 0 to 200. A fuzzy set, *Young*, may be defined by the function

$$\mu_{Young}(x) = \begin{cases} 1 & 0 \le x \le 25; \\ (1 + (\frac{x-25}{5})^2)^{-1} & 25 < x \le 200. \end{cases}$$

Thus, the age 25 is definitely young; the age 27 is young to a degree of 0.86; and the age 60 is young to a degree of 0.02.

Standard (or crisp) sets are special fuzzy sets whose membership function maps every element in the set to degree 1 and all others to degree 0. Set operations, such as union, intersection, etc., are also extended to fuzzy sets. For example, if an element $a$ is in fuzzy sets $S_1$ with degree 0.4 and in $S_2$ with degree 0.75, then $a$ is in $S_1 \cup S_2$ with a degree $max(0.4, 0.75) = 0.75$ and in $S_1 \cap S_2$ with a degree $min(0.4, 0.75) = 0.4$. Readers interested in the fuzzy sets theory may refer to [18, 19].

## 2.2  IMPRECISE QUERY

An *imprecise (or fuzzy) query* is formulated using linguistic terms whose meaning is imprecise. Consider a corporation database containing information about employees, departments, etc. Some imprecise queries may be the following.

"List names of employees who are young and well–paid."

"List names of departments in which most employees are young."

In these queries, "young", "well–paid", and "most" are linguistic terms, and have the following characteristics.

1. They are imprecise. For example, it may not be clear whether the age 32 is young or not young.

2. They are subjective and context dependent. For example, for employees, the age 19 may be definitely young, for children of employees, the age 19 may be definitely not young.

Linguistic terms used in imprecise queries can be defined using fuzzy sets and may be classified into four types.

1. Simple fuzzy concepts, such as, "young", "well–paid", "about 25", etc.

2. Fuzzy modifiers, such as, "very", "much", "more or less", "a little bit", etc.

3. Fuzzy relationships, such as, "likes", "similar to", "close to", "far apart", etc.

4. Fuzzy quantifiers, such as, "most", "almost", "a few", etc.

The formal query languages for fuzzy databases that appeared in the literature support the first three types of linguistic terms, but not fuzzy quantifiers. The languages in this paper will support all four types.

# 3  FUZZY RELATIONS

In this section, we define the fuzzy relation which is an extension of a standard relation [7, 11, 17] using the concept of fuzzy sets. In our presentation, the standard terminology of relational database as defined in [16] is used.

**Definition 3.1** A fuzzy relation scheme $F = (A_1, \ldots, A_n)$ is a set of $n$ distinct attributes. The domain of $A_i$, denoted by $DOM(A_i)$, is a set of atomic, precise values. The *domain* of $F$, denoted by $DOM(F)$, is the set $\{< a_1 \ldots a_n >| a_i \in DOM(A_i)\}$. A *fuzzy relation* $r$ with scheme $F$, denoted by $r(F)$ (or simply $r$ when the scheme is understood), is a (sub)set of $DOM(F)$ defined by a membership function $\mu_r : DOM(F) \rightarrow [0,1]$, such that a tuple $t$ in $DOM(F)$ is in $r(F)$ iff $\mu_r(t) > 0$, where $\mu_r(t)$ is the *degree of $t$ wrt $r$*. A fuzzy database is a finite set of fuzzy relations each of which is with a fuzzy relation scheme. □

By this definition, a standard (or crisp) relation is a special fuzzy relation whose membership function assigns a degree 1 to every tuple in the relation and a degree 0 to every tuple not in the relation. Similarly, a standard relational database is a special fuzzy relational database in which every relation is crisp. Unlike the standard relations, for two fuzzy relations with the same scheme to be the same, the two fuzzy relations not only must have the same set of tuples, but also must have the same membership functions.

Our definition of fuzzy relations differs from those in [2, 3, 4, 5, 12, 14, 20] in that the domains of attributes in our definition contain only atomic, precise values while the domains of attributes in their definitions may contain fuzzy linguistic values, represented by possibility distributions, and the null value.

As an example, Figure 1 contains a scheme of a fuzzy relational database that will be referenced in examples appearing in this paper.

$Employee = (Eid, Name, Addr, Age, title, Specialty, Sal, DNo)$
$Department = (D\#, Name, Addr, Chair)$
$Likes = (Subj, Obj)$
$CloseTo = (Addr1, Addr2)$
$Young = (Age)$
$WellPaid = (Sal)$
$Most = (Degree)$
$Very = (Degree)$

Figure 1: An Example Fuzzy Relational Database Scheme.

# 4 A FUZZY TUPLE RELATIONAL CALCULUS

In this section, we present a fuzzy tuple relational calculus based on the fuzzy relations.

## 4.1 THE SYNTAX

A fuzzy tuple relational calculus expression is defined by $R = \{t \mid \psi(t)\}$, where $t$ is a tuple variable, and $\psi(t)$ is a fuzzy logic formula. The expression denotes the set of tuples that satisfy formula $\psi(t)$ with a degree in $[0,1]$.

The fuzzy logic formula is similar to the standard one with two extensions. First, the membership degree of tuples wrt fuzzy relations can be used to select (see case 4 of the basis in Definition 4.1) and to connect (see case 5 of the basis in Definition 4.1) tuple variables. This allows a variable degree of fuzziness to be specified when selecting tuples from relations, and allows the fuzziness of tuples in a fuzzy relation to be modified by the fuzziness of tuples in another fuzzy relation. The latter feature is useful for representing the modification of a linguistic term by another linguistic term, as in "very young" (see Example 4.2 for more details). Second, two types of fuzzy quantifiers are allowed. The first quantifier allows the specification of sentences such as "most (almost all, a few, ...) of $t$'s that satisfy condition 1 (with a degree greater than 0) satisfy the condition 2 (with a degree greater than 0)" (see case 6 of the induction in Definition 4.1). For example, the sentence

"Most tall men are not very fat" is of this type. where "tall men" is the condition 1. and "not very fat (men)" is the condition 2. The second quantifier allows the specification of sentences such as "most (almost all. a few, ...) of the following ($k$) conditions are satisfied (with degrees greater than 0)" (see case 7 of the induction in Definition 4.1). Fuzzy quantifiers are linguistic terms representing unary fuzzy relations whose only attribute has the domain [0.1]. In the sequel. $t[A]$ denotes the component of a tuple variable $t$ under attribute $A$.

**Definition 4.1** Let $r$ be a fuzzy relation; $t$ and $v$ be tuple variables; $A$ and $B$ be attributes; $c$ and $k$ be constants, where $k$ is in $[0,1]$; $\mu_r(t)$ be the degree of $t$ wrt $r$; and $\theta \in \{=,\neq,<,\leq,>,\geq\}$ be a comparison operator.

 **Basis:** Any one of the following is a *fuzzy logic formula* (or simply a *formula*):

1. $t \in r$.

2. $t[A]\ \theta\ c$ (or $c\ \theta\ t[A]$).

3. $t[A]\ \theta\ v[B]$.

4. $\mu_r(t)\ \theta\ k$.

5. $\mu_r(t)\ \theta\ v[B]$, where $v[B] \in [0,1]$.

 **Induction:** Let $E_1, E_2, \ldots, E_k$ be fuzzy logic formulas. Then, the following are also fuzzy logic formulas.

1. $\neg E_1$, where $\neg$ is logic negation.

2. $E_1 \wedge E_2$, where $\wedge$ is logic **AND**.

3. $E_1 \vee E_2$, where $\vee$ is the logic **OR**.

4. $(\forall t)(E_1(t))$, where $\forall$ is a universal quantifier.

5. $(\exists t)(E_1(t))$, where $\exists$ is an existential quantifier.

6. $(F\ t : E_1(t))(E_2(t))$, where $F$ is a fuzzy quantifier.

7. $(F\ (E_1, \ldots, E_k))$, where $F$ is a fuzzy quantifier.

8. $(E)$, where () is used to change the priority of evaluating a sub-formula. The priority of the connectives is given by $\neg, \wedge, \vee$, in that order.

9. Nothing else is a formula. $\square$

Similar to the standard tuple relational calculus, the subset containing operators $\neg E$, $E_1 \wedge E_2$, $(E)$, $(\exists t)(E(t))$, $(F\ t : E_1(t))(E_2(t))$, and $F\ (E_1, \ldots E_k)$ is sufficient for formulating any expressions in the calculus. The remaining operators can be obtained as follows.

1. $E_1 \vee E_2 \equiv \neg(\neg E_1 \wedge \neg E_2)$.

2. $\forall t(E) \equiv \neg \exists t(\neg E)$.

## 4.2 THE SEMANTICS

The (fuzzy) truth value of a fuzzy logic formula is a real value in $[0, 1]$, where $0$ representing "definitely false", $1$ representing "definitely true", and other values representing various degree of "partially true". To define the truth value of a formula, let $Truth()$ be a function that maps a fuzzy logic formula to its truth value. We also define two operators[4], $\odot$ and $\div$, on the set of real numbers in $[0, 1]$ for computing the fuzzy truth values of conjunctive and disjunctive formulas, respectively. These operators have the following properties.

1. $a \odot 0 = 0, a \odot 1 = a$;

2. $0 \le a \odot b \le 1$;

3. $a \otimes b = b \otimes a$;

4. $a \otimes b \otimes c = a \otimes (b \odot c) = (a \otimes b) \odot c$.

5. $a \otimes b \ge c \otimes d$ if $a = c$ and $b \ge d$, or $a \ge c$ and $b = d$.

6. $a \oplus 1 = 1, a \oplus 0 = a$;

7. $0 \le a \oplus b \le 1$;

8. $a \oplus b = b \oplus a$;

9. $a \oplus b \oplus c = a \oplus (b \oplus c) = (a \oplus b) \oplus c$.

10. $a \oplus b \ge c \oplus d$ if $a \ge max(c, d)$ or $b \ge max(c, d)$.

11. $a \otimes b \equiv 1 - ((1 - a) \oplus (1 - b))$.

12. $a \oplus b \equiv 1 - ((1 - a) \otimes (1 - b))$.

A popular choice of these two operators is that $\otimes \equiv min$ and $\oplus \equiv max$.

The truth value of a given fuzzy logic formula is obtained from that of sub-formulas based on following rules.

1. For any instantiation of $t$, $Truth(t \in r) = \mu_r(t)$.

2. For any instantiation of $t$ and any constant $c$, $Truth(t[A] \ \theta \ c) = 1$ if the condition $t[A] \ \theta \ c$ holds, and $0$ otherwise. The truth value of $t[A] \ \theta \ v[B]$, $\mu_r(t) \ \theta \ k$, and $\mu_r(t) \ \theta \ v[B]$ are defined similarly. That is, the truth values of these basic formulas are binary.

3. For any formula $E$, $Truth(\neg E) = 1 - Truth(E)$.

4. For any two formulas $E_1$ and $E_2$, $Truth(E_1 \vee E_2) = Truth(E_1) \oplus Truth(E_2)$; and $Truth(E_1 \wedge E_2) = Truth(E_1) \otimes Truth(E_2)$.

5. for any formula $E$, $Truth((E)) = Truth(E)$.

6. For any formula $E$, $Truth((\forall t)(E(t))) = Truth(E(t_1)) \otimes \cdots \otimes Truth(E(t_n))$, where $t_1, \ldots, t_n$ are all possible instantiations of $t$.

7. For any formula $E$, $Truth((\exists t)(E(t))) = Truth(E(t_1)) \oplus \cdots \oplus Truth(E(t_n))$, where $t_1, \ldots, t_n$ are all possible instantiations of $t$.

---

[4]In the literature, $\otimes$ is also called the $t$–norm; $\oplus$ is also called the $t$–conorm or the $s$–norm.

8. For any fuzzy quantifier $F$ and any formula $E_1$ and $E_2$.

$$Truth((F\ t : E_1(t))(E_2(t))) = \mu_F(\phi)$$

where $\phi = (\sum_{i=1}^{k} Truth(E_1(t_i) \wedge E_2(t_i)))/(\sum_{j=1}^{k} Truth(E_1(t_j)))$; $t_1, \ldots, t_k$ are all possible instantiations of $t$ for which $Truth(E_1(t_i)) > 0$; and $\mu_F$ is the membership function of the fuzzy quantifier $F$. Intuitively, $\phi$ indicates the average possibility for $t_i$ to satisfy both $E_1$ and $E_2$, given that $t_i$ satisfies $E_1$, where the satisfaction is in terms of a degree. We assume that $\phi = 0$ is $Truth(E_1(t_i)) = 0$ for all $i$. Example 4.4 provides further explanation about this type of quantification.

9. For any fuzzy quantifier $F$ and any $k$ formulas $E_1, \ldots E_k$.

$$Truth(F\ (E_1, \ldots, E_k)) = \mu_F((Truth(E_1) + \cdots + Truth(E_k))/k)$$

where $\mu_F$ is the membership function of the fuzzy quantifier $F$. Intuitively, the truth value of $F(E_1, \ldots, E_k)$ is the membership degree of the average truth value of $E_i$ with respect to the fuzzy quantifier $F$. More explanations are given in Example 4.5.

If all relations are crisp, the truth values will become binary, and rules 1 through 7 are exactly the same as that in the standard tuple relational calculus. On the other hand, rules 8 and 9 are unconventional.

## 4.3  SPECIFY QUERIES IN THE CALCULUS

Following examples illustrates the use of the fuzzy tuple relational calculus.

**Example 4.1** "Find names of employees who are young with a degree greater than 0.75".

$Answer = \{t \mid (\exists e, y)(e \in Employee \wedge y \in Young \wedge t[Name] = e[Name]$

$\wedge e[Age] = y[Age] \wedge \mu_{Young}(y) > 0.75)\}$

This example shows the use of simple fuzzy concepts $Employee$[5] and $Young$, and the use of the membership degree as a threshold to select tuples for the answer. For any tuple $e$ in $Employee$ and any tuple $y$ in $Young$, the truth value of the formula is 0 if any one of the conjuncts has truth value 0. Otherwise the truth value is determined only by $\mu_{Employee}(e) \otimes \mu_{Young}(y)$, since all other conjuncts would have had the truth value 1. If $\mu_{Employee}(e) = 1$, then the degree for $t$ to be in the answer is the degree for $t$ to be the name of a employee whose age is considered young with a certainty higher than 0.75. □

**Example 4.2** "Find names and titles of all employees who are very young and well-paid".

$Answer = \{t \mid (\exists e, v, y, w)(e \in Employee \wedge y \in Young \wedge w \in WellPaid \wedge$

$v \in Very \wedge t[Name] = e[Name] \wedge t[Title] = e[Title] \wedge$

$e[Age] = y[Age] \wedge \mu_{Young}(y) = v[Degree] \wedge e[Sal] = w[Sal])\}$

This example illustrates the use of fuzzy modifiers such as the term "very". For a given set of tuples $e$, $v$, $y$, and $w$, the truth value of the conjunction in the formula is determined as follows. If any conjunct has a truth value 0, the truth value of the conjunction is also 0. If all conjuncts have truth values greater than 0, the truth value of the conjunction is computed by $\mu_{Employee}(e) \otimes \mu_{Young}(y) \otimes \mu_{Very}(v) \otimes \mu_{WellPaid}(w)$. If the term "very" does not appear in the query, the truth value would be computed by $\mu_{Employee}(e) \otimes \mu_{Young}(y) \otimes \mu_{WellPaid}(w)$ which evaluates to a higher

---

[5] As a special case, $Employee$ may be a crisp relation.

truth value. This is natural since "very young" is a stronger condition than "young" and therefore is more difficult to satisfy. Terms such as "very" can be used to modify many other terms such as "large", "beautiful", etc., and the content in their fuzzy relations are determined based on $\odot$ operator. For example, if $\odot$ is the operator *min*, a common choice of the membership function of "very" is $\mu_{Very}(x) = x^2$. Threshold values can be used in this query to fine-tune the fuzziness. For example, the query can be "Find names and titles of individuals who satisfy the following criteria a a degree higher than 0.73: they are employees who are very young to a degree higher than 0.55 and well-paid to a degree higher than 0.85". The threshold values can be incorporated into the query by adding $\mu_{Answer}(t) > 0.73 \land \mu_{Very}(v) > 0.55 \land \mu_{WellPaid}(w) > 0.85$ into the fuzzy logic formula. The higher threshold value for *Well-paid* implies that being well-paid is a stronger condition than being very young. □

**Example 4.3** "Find names of employees who live close to work".

$Answer = \{t \mid (\exists e, c, l)(e \in Employee \land d \in Department \land l \in CloseTo \land$

$t[Name] = e[Name] \land e[Addr] = l[Addr1] \land d[Addr] = l[Addr2])\}$

This example illustrates the use of a fuzzy relationship "close to". The fuzzy relation *CloseTo* describes the similarity between pairs of addresses, thus is used to connect employees with departments. In fact, the degree of a tuple in *CloseTo* can be thought as the degree of similarity of the two components of the tuple. □

**Example 4.4** "Find names of departments in which most employees are young."

$Answer = \{t \mid (\exists d)(d \in Department \land t[Name] = d[Name] \land$

$(Most\ e : (e \in Employee \land e[Dno] = d[D\#]))((\exists y)(y \in Young \land e[Age] = y[Age])))\}$

Here, we illustrate the use of the first of the two types of fuzzy quantifiers. In this query, the fuzzy quantifier "most" is associated with a subset of employees — those who are in the same department — rather than all possible tuples under the scheme of *Employee*, as in the cases of "forall" and "exists". Conceptually, for each department, we consider each employee in that department to determine his/her degree of being young, and then measure the department wrt the "most" based on the departmental average degree of being "young". Notice that by the definition of the truth value, for an employee to be considered, he/she must be in the given department with a degree greater than 0, but his/her degree of being young is allowed to be 0. □

**Example 4.5** "Find the name of employees who meet most of the following criteria: young, well-paid, and interested in traveling."

$Answer = \{t \mid (\exists e)(e \in Employee \land t[Name] = e[Name] \land$

$(Most\ ((\exists y)(y \in Young \land e[Age] = y[Age]),$

$(\exists w)(w \in WellPaid \land e[Sal] = w[Sal]),$

$(\exists l)(l \in Likes \land l[Subj] = e[Eid] \land l[Obj] = "Traveling")))\}$

This example illustrates the use of the second type of the fuzzy quantifiers. There are three conditions to be qualified by tuples in the answer, but not all three must be satisfied to 100 percent. For each employee, the average degree with which all three conditions are satisfied is measured against the quantifier "most". □

# 5 A FUZZY RELATIONAL ALGEBRA

In this section, we define a fuzzy relational algebra based on the fuzzy relations. Like the standard relational algebra, the fuzzy relational algebra consists of a set of operations on fuzzy relations. These operations can be divided into two groups, the basic ones and the additional ones.

## 5.1 BASIC OPERATIONS

The set of basic operations in the fuzzy relational algebra are defined below. In the following, let $r(R)$ and $s(S)$ be two fuzzy relations.

The *Cartesian product* of $r$ and $s$ denoted by $r \times s$, is a fuzzy relation $\epsilon(RS)$, where $RS$ is the concatenation of $R$ and $S$. The relation $\epsilon$ contains the set of tuples obtained by pairing each tuple in $r$ with each tuple in $s$. For each tuple $t$ in $\epsilon$, suppose that $t$ is obtained from $t_1$ in $r$ and $t_2$ in $s$, then the degree of $t$ in $\epsilon$ is $\mu_e(t) = \mu_r(t_1) \odot \mu_s(t_2)$.

We say that $r$ and $s$ are *union compatible* if $R = (A_1, \ldots, A_n)$ and $S = (B_1, \ldots, B_n)$, and for each $1 \leq i \leq n$, $DOM(A_i) = DOM(B_i)$.

Let $r$ and $s$ be union compatible fuzzy relations. The *union* of $r$ and $s$, denoted by $r \cup s$, is a fuzzy relation $e(R)$ (or $e(S)$) which contains all tuples that are either in $r$ or in $s$. For each tuple $t$ in $\epsilon$, the membership degree of $t$ wrt $\epsilon$ is $\mu_e(t) = \mu_r(t) \oplus \mu_s(t)$.

The *intersection* of $r$ and $s$, denoted by $r \cap s$, is a fuzzy relation $\epsilon(R)$ (or $\epsilon(S)$) which contains all tuples that are in both $r$ and $s$. For each tuple $t$ in $\epsilon$, the membership degree of $t$ wrt $\epsilon$ is $\mu_e(t) = \mu_r(t) \odot \mu_s(t)$.

The *set-difference* of $r$ and $s$, denoted by $r - s$, is a fuzzy relation $e(R)$ which contains all tuples that are in $r$ but are not in $s$ with degree 1. For each tuple $t$ in $\epsilon$, the membership degree of $t$ wrt $e$ is $\mu_e(t) = \mu_r(t) \otimes (1 - \mu_s(t))$.

The selection operation is based on a *conditional formula* which is either an atomic formula or a more complex formula formed from simpler conditional formula using **and** ($\wedge$) and **or** ($\vee$). The atomic conditional formulas are of the following form: $r.A \; \theta \; c$, $r.A \; \theta \; s.B$, $\mu_r() \; \theta \; k$, and $\mu_r() \; \theta \; s.B$, where $r.A$ denotes the attribute $A$ of fuzzy relation $r$; $c$ and $k \in [0,1]$ are constants; $\theta \in \{=, \neq, <, \leq, >, \geq\}$; and $\mu_r()$ denotes the membership function of fuzzy relation $r$. Notice that conditional formulas are ordinary logic formulas, that is their truth values are binary.

The *selection* of $r$ based on a conditional formula $Q$, denoted by $\sigma_Q(r)$, is a fuzzy relation $e(R)$ which contains all tuples in $r$ that satisfy $Q$. For each tuple $t$ in $e(E)$, the membership degree of $t$ wrt $e$ is $\mu_e(t) = \mu_r(t)$.

Let $L$ be a sequence of attributes of scheme $R$. The *projection* of $r$ on $L$, denoted by $\Pi_L(r)$, is a fuzzy relation $e(L)$. There is a tuple $t$ in $e$ if there are $k \geq 1$ tuples $t_1, \ldots, t_k$ in $r$ with membership degrees $\mu_r(t_1), \ldots, \mu_r(t_k)$, respectively, such that for every attribute $A$ in $L$, $t[A] = t_1[A] = \cdots = t_k[A]$. The degree of $t$ in $e$ is $\mu_e(t) = \mu_r(t_1) \oplus \cdots \oplus \mu_r(t_k)$. Intuitively, the projection is the same as that in the standard relational algebra except that the duplicate elimination process must also take care of the calculation of the membership degree. That is, before removing duplicate tuples, each tuple carries over the membership degree of its original tuple in the operand relation $r$, and when the duplicate is removed, the membership degrees of all tuples that are the same on all attributes in $L$ are used to compute the membership degree wrt the resulting relation.

The *renaming* of a relation $r(R)$ to $e(E)$, denoted by $e(E) \leftarrow r$, is the same relation as that of $r$, except that the name of the relation, and maybe the names of the attributes, are changed.

The above algebraic operations will reduce to those in the standard relational algebra when all operands are crisp relations. The next two operations are special to the fuzzy relational algebra and are motivated to match directly the two types of fuzzy quantifiers of the fuzzy tuple relational calculus.

Let $r$ and $s$ be union-compatible fuzzy relations, $L$ be a subset of attributes in scheme $R$, and $F$ be a unary fuzzy relation whose only attribute has the domain $[0,1]$. The $\Omega$-*mapping* of $r$ and $s$ with respect to $L$ and $F$, denoted by $r \rhd_L^F s$, is a fuzzy relation $e(L)$ which is obtained as follows.

1. Group the tuples in $r$ by attributes $L$.

2. Compute $r \cap s$ and group tuples in $r \cap s$ by attributes $L$.

3. Compute a fuzzy relation $e_1(L)$ as follows. For each group in $r \cap s$, let the value (sub-tuple) for the attributes $L$ be $l_i$. Then $e_1$ contains a tuple $l_i$ with a membership degree equals the

sum of the membership degrees of the group $l_i$ in $r \cap s$ divided by the sum of the membership degrees of the group $l_i$ in $r$.

4. Let $\epsilon$ be the same as $\epsilon_1$ except for each tuple $l_i$ in $\epsilon_1$, $\mu_\epsilon(l_i) = \mu_F(\mu_{\epsilon_1}(l_i))$.

The Example 5.4 in Section 5.3 illustrates the use of this operation.

Let $r_1(R_1), \ldots, r_k(R_k)$ be $k \geq 1$ union-compatible fuzzy relations and $F$ be a unary fuzzy relation whose only attribute has the domain $[0, 1]$. The $\Sigma$-intersection of $r_1, \ldots, r_k$ with respect to $F$, denoted by $\Sigma_F(r_1, \ldots, r_k)$, is a fuzzy relation $\epsilon(R_1)$ (or $\epsilon(R_i)$, $1 \leq i \leq k$). There is a tuple $t$ in $\epsilon$ if $t$ is in at least one $r_i$, for $1 \leq i \leq k$. For each $t$ in $\epsilon$, the membership degree of $t$ is $\mu_\epsilon(t) = \mu_F((\mu_{r_1}(t) + \cdots + \mu_{r_k}(t))/k)$. The use of this operation is illustrated by Example 5.5 in Section 5.3.

The formal definition of the fuzzy relational algebra is given below.

**Definition 5.1** An expression in fuzzy relational algebra is defined inductively as follows.

1. A fuzzy relational variable is a fuzzy relational expression.

2. If $E_1, \ldots, E_k$ are fuzzy relational expressions. Then, so are

    (a) $E_1 \times E_2$.

    (b) $E_1 \cup E_2$.

    (c) $E_1 \cap E_2$.

    (d) $E_1 - E_2$.

    (e) $\sigma_Q(E_1)$.

    (f) $\Pi_L(E_1)$.

    (g) $E_1 \triangleright_L^F E_2$.

    (h) $\Sigma_F(E_1, \ldots, E_k)$.

    (i) $r(R) \leftarrow E_1$, where $r(R)$ is a fuzzy relation with scheme $R$. □

## 5.2 ADDITIONAL OPERATIONS

The set of basic fuzzy algebraic operations are sufficient for specifying any expression in the fuzzy relational algebra. But it is more convenient to define additional operations, as in the standard relational algebra.

Let $r$ and $s$ be fuzzy relations as defined before, and $Q$ be a conditional formula. The $\theta$-join of $r$ and $s$, denoted by $r \bowtie_Q s$, is a fuzzy relation $\epsilon(RS)$ where $RS$ is the concatenation of $R$ and $S$. A tuple $t$ is in $\epsilon$ if there is a tuple $t_1$ in $r$ and a tuple $t_2$ in $s$, such that $t[R] = t_1[R]$, $t[S] = t_2[S]$, and $t_1$ and $t_2$ together satisfy $Q$. The degree of $t$ in $\epsilon$ is $\mu_\epsilon(t) = \mu_r(t_1) \otimes \mu_s(t_2)$.

It is obvious that the $\theta$-join can be obtained by Cartesian product followed by a selection, that is, $r \bowtie_Q s = \sigma_Q(r \times s)$.

The *equijoin* of $r$ and $s$ is the same as the $\theta$-join of $r$ and $s$ except that $Q$ contains only equalities. The *natural join* of $r$ and $s$, denoted by $r \bowtie s$, is also the same as its counterpart in the standard relational algebra except that each tuple in the result has a membership degree computed as in the $\theta$-join.

Notice that unlike the standard relational algebra, the operation $r \cap s$ may not be equivalent to $r - (r - s)$. To see this, assume that the membership degree of a tuple $t$ wrt $r$ is $a$ and that wrt $s$ is $b$. By definition, the membership degree of $t$ wrt $r \cap s$ is $a \otimes b$, and that of $t$ wrt $r - (r - s)$ is $a \otimes (1 - (a \otimes (1 - b)))$. Thus the two expressions are equivalent iff $a \otimes b = a \otimes (1 - (a \otimes (1 - b)))$. If $\otimes \equiv min$, the two expressions are equivalent. But if $\otimes \equiv \times$, they are not equivalent.

## 5.3 SPECIFY QUERIES IN THE ALGEBRA

We now express the set of queries appeared in examples in Section 4.3, using the fuzzy relational algebra.

**Example 5.1** "Find names of employees who are young with a degree greater than 0.75".

$$\Pi_{Name}(Employee \bowtie (\sigma_{\mu_{Young}()>0.75} Young))$$

In this example, the natural join is on the attribute $Age$ in both $Employee$ and $Young$. The threshold value 0.75 is used first to select the ages from the fuzzy relation $Young$. The membership degree of tuple $t$ in the answer is computed by $\mu_{Employee}(t) \odot \mu_{Young}(t.Age)$. □

**Example 5.2** "Find names and titles of all employees who are very young and well-paid".

$$\Pi_{Name, Title}(Employee \bowtie (Young \bowtie_{\mu_{Young}()=Very.Degree} Very) \bowtie WellPaid)$$

Notice that the modification of $Young$ by $Very$ is obtained by an equijoin where one side of the join condition involves the membership function of $Young$. Again, various threshold values can be used to select tuples from the operand fuzzy relations that participate the joins. For example, the query "Find names and titles of individuals who satisfy the following criteria to a degree higher than 0.73: they are employees who are very young to a degree higher than 0.55 and well-paid to a degree higher than 0.85" can be expressed as

$$\sigma_{\mu_R()>0.73}(R \leftarrow (\Pi_{Name, Title}((Young \bowtie_{\mu_{Young}()=Very.Degree} (\sigma_{\mu_{Very}()>0.55}Very) \bowtie$$
$$Employee \bowtie (\sigma_{\mu_{WellPaid}()>0.85}WellPaid))))\square$$

**Example 5.3** "Find names of employees who live close to work".

$$\Pi_{Name}(Employee \bowtie_{Employee.Addr=CloseTo.Addr1} CloseTo$$
$$\bowtie_{Department.Addr=CloseTo.Addr2} Department)\square$$

**Example 5.4** "Find names of departments in which most employees are young."

$$\Pi_{Department.Name}(Department \bowtie_{Employee.Dno=Department.D\#}$$
$$(Employee \triangleright^{Most}_{Employee.Dno} (Employee \bowtie Young)))$$

the sub-expression involving the $\Omega$-mapping is computed as the follows.

1. For each department number in $Employee$, the sum of the membership degree of all employees in that department is computed.

2. For each department number in $Employee \bowtie Young$, the sum of the membership degree of all young employees in that department is computed.

3. For each department number in $Employee \bowtie Young$, the ratio of the sum obtained in step 2 to that obtained in step 1 is obtained.

4. The resulting fuzzy relation contains distinct department numbers in $Employee \bowtie Young$, and for each department number, the membership degree is that of the corresponding ratio computed in step 3 with respect to the fuzzy relation $Most$. □

**Example 5.5** "Find names of employees who meet most of the following criteria: young, well-paid, and interested in traveling."

$$\Sigma_{Most}((\Pi_{Employee.Name}(Employee \bowtie Young)),$$
$$(\Pi_{Employee.Name}(Employee \bowtie WellPaid)),$$
$$(\Pi_{Employee.Name}(Employee \bowtie_{Employee.Eid=Likes.Subj} (\sigma_{Likes.Obj="Traveling"}Likes))))$$

The three sub-expressions compute the young employees, the well-paid employees, and the travel-liking employees, respectively. The resulting fuzzy relation contains the name of every employee who is in any one of the three categories. For each employee name in the result, the membership degree is computed by first find the average membership degree with which the employee falls into all three categories, and then find the membership degree of this average wrt the fuzzy relation *Most*. □

# 6    SAFETY AND EQUIVALENCE

Like in the standard tuple relational calculus, not every fuzzy tuple relational expression is useful. For example, if a fuzzy tuple calculus expression denotes an infinite fuzzy relation, there is no way to obtain all tuples in the fuzzy relation, and there may be no way to finitely represent the fuzzy relation, neither. In the following, an expression is said to be *infinite* if it denotes an infinite fuzzy relation. In the fuzzy tuple relational calculus, there are two situations in which an expression may be infinite. In the first situation, the formula involves infinite fuzzy relations, but does not restrict them. For example, consider a unary fuzzy relation, say *FarAway* with an attribute *Distance* whose domain is the set of integers greater than or equal to 0 (representing kilometers). Assume that the membership function of *FarAway* is defined by

$$\mu_{FarAway}(t) = \begin{cases} 0, & 0 \le t \le 2; \\ (1 + (\frac{2}{t-2})^2)^{-1}, & t > 2 \end{cases}$$

Then, the expression $\{t \mid t \in FarAway\}$ represents the query "List all tuples in *FarAway*" and denotes an infinite fuzzy relation. In the second situation, the operators such as $\neg$ and $\forall$ may force the tuples to be formed using symbols that are neither in the database nor in the given query. For example, $\{t \mid \neg(t \in r \wedge \mu_r(t) = 1)\}$ may be infinite. In the standard tuple relational calculus, a notion of safety was defined, and only safe expressions are of practical interests. Intuitively, a safe tuple calculus expression denotes a finite relation in which each tuple is formed using only those symbols that are either in the database relevant to the query or in the query itself. In the following, we extend the notion of safety to fuzzy tuple relational calculus expression.

**Definition 6.1** Given a fuzzy logic formula $\psi$. Let $SYM(\psi)$ be the set of symbols that either appear explicitly in $\psi$ or are components of some tuples that are in some fuzzy relation mentioned in $\psi$. □

Intuitively, $SYM(\psi)$ is the set of symbols that are in the relevant portion of the database and in the given query. Since fuzzy relations may be infinite, $SYM(\psi)$ may also be infinite. In the following, we say that a variable is *bounded* if it is associated with a quantifier, and it is *free*, otherwise.

**Definition 6.2** Let $R = \{t \mid \psi(t)\}$ be a fuzzy tuple relational calculus expression. The formula $\psi(t)$ is said to be *range restricted* if

1. Whenever $t$ satisfies $\psi(t)$ with a non-zero degree, each component of $t$ is in $SYM(\psi)$.

2. For each sub-formula of $\psi(t)$ of the form $(\exists u)(\omega(u))$, if $u$ satisfies $\omega(u)$ with a non-zero degree for any free variable in $\omega(u)$, then each component of $u$ is in $SYM(\omega)$.

3. For each sub-formula of $\psi$ of the form $(\forall u)(\omega(u))$, if any component of $u$ is not in $SYM(\omega)$, then $u$ satisfies $\omega$ with the degree 1. Intuitively, the quantifier "forall" requires checking of every tuple in the domain of $u$ for the satisfaction of $\omega$. But by this definition, tuples with components formed using symbols outside of $SYM(\omega)$ need not be checked, since they will never affect the truth value of the sub-formula.

4. For each sub-formula of $\psi$ of the form $(F\ u : \omega_1(u))(\omega_2(u))$, if any component of $u$ is not in $SYM(\omega_1)$ then $u$ satisfies $\omega_1$ with the degree 0. Since only those $u$'s that satisfy $\omega_1$ with a

non-zero degree will contribute to the truth value of this sub-formula. this definition simply says that there is no need to check tuples which have at least one component with a symbol outside of $SYM(w_1)$. since such a tuple is automatically out of the consideration.

The fuzzy tuple relational calculus expression is *range restricted* if the formula $v(t)$ is range restricted. □

Intuitively, if a fuzzy tuple relational calculus expression is range restricted. every component of every tuple in the answer is a symbol in the database or in the query. Therefore, one only needs to search for the answer within the set of tuples that can be constructed using symbols in $SYM(v)$. In the standard tuple relational calculus, the range restriction alone is sufficient in defining the safety of the expressions. But in the fuzzy tuple relational calculus, a range restricted expression may still not necessarily denote a finite fuzzy relation since $SYM(v)$ may be infinite. Thus, we need the following more general definition of the safety.

**Definition 6.3** An expression in the fuzzy tuple relational calculus is *finite* if it denotes a finite fuzzy relation and it is *safe* if it is range restricted and finite. □

For the fuzzy relational algebra, we define a *finite* expression to be the one that evaluates to a finite fuzzy relation.

The following two theorems together state that the fuzzy tuple relational calculus and the fuzzy relational algebra are equivalent in terms of their expressive power. Due to the space limitation, these theorems are provided without proof. Readers who are interested may refer to [21].

**Theorem 6.1** If $E$ is a expression in the fuzzy relational algebra, there is a range restricted expression in the fuzzy tuple relational calculus equivalent to $E$. Furthermore, if $E$ is finite, the equivalent expression in the fuzzy tuple relational calculus is safe. □

**Theorem 6.2** Each range restricted expression in the fuzzy tuple relational calculus has an equivalent expression in the fuzzy relational algebra. Furthermore, if the fuzzy tuple relational calculus expression is safe, the equivalent algebraic expression is finite. □

# 7 IMPLEMENTATION CONSIDERATIONS

In this section, we briefly discuss techniques that allow an implementation of the proposed data model. More specifically, we discuss the implementation of fuzzy relations in a standard relational database, the user interface, and the structure of a front–end system.

In the fuzzy relational database, there are two types of fuzzy relations: the finite ones and the infinite ones. The implementation of these two types of fuzzy relations is naturally different.

Each finite fuzzy relation can be represented by a standard relation which contains a designated attribute for the membership degree. This attribute can be made accessible to the user so that the user may inspect or change the membership degree of any tuple. This allows a fuzzy relation to represent the personal viewpoints of the user about the fuzzy concept represented by the fuzzy relation.

There may be several ways to implement infinite fuzzy relations depending on the types of their membership functions. For instance, a single attribute fuzzy relation with a membership function $\mu_r(x) = x^2$ over integers is probably best implemented as an ordinary function. A fuzzy data dictionary (or library) can be maintained and used by a front–end system to provide the mapping between the fuzzy relation and the function that implements its membership function. The fuzzy data dictionary itself may partially be stored in the standard relational database as well. For example, if several infinite fuzzy relations have similar membership functions. There is no need to implement a distinct function for each of these fuzzy relations. Instead, a generic function can be implemented which when supplied with appropriate values of parameters can implement the

membership function of any one of these fuzzy relations. The set of values of the parameters for different fuzzy relations must be in the fuzzy data dictionary. and can be stored in a standard relation.

The fuzzy relational algebraic operations can be implemented in two steps. In the first step. the standard relational algebraic operations are called for to perform the set-oriented retrieval operations. In the second step. a post processing is required to perform operations that can not be accomplished by the standard relational operations. such as joining with infinite fuzzy relations and calculating the membership degrees of the resulting tuples. For example. to compute the answer to the query in Example 5.3. the standard relational joins can be performed to obtain the join of $Employee$, $CloseTo$. and $Department$. The resulting intermediate relation is necessary to have three designated attributes for $\mu_{Employee}()$, $\mu_{CloseTo}()$. and $\mu_{Department}()$. respectively. Then. the intermediate relation can be scanned to generate the final answer, and during this step, the membership degree of each tuple in the final answer is computed using values in the three designated attributes.

The user interface provides facilities to allow a user to query the database and to manipulate the fuzzy relations. The fuzzy tuple relational calculus and the fuzzy relational algebra do not provide all functionalities that are provided by the standard relational database languages, such as SQL. A fuzzy SQL can be provided as the user interface to the fuzzy relational database which is based on our model. This fuzzy SQL can allow linguistic terms. such as $Young$, $WellPaid$, to be used in the $where$-clause as a constant. For example, the query in Example 4.2 can be expressed as

```
SELECT Name, Title
FROM   Employee
WHERE  Age = Very Young AND Sal = WellPaid
```

The basic structure of a fuzzy database system consists of a fuzzy SQL based user interface, a fuzzy data (function) library, a front-end system that performs the two-step implementation of fuzzy relational algebraic operations, and a standard relational database.

# 8   CONCLUSION

In this paper, we propose a fuzzy relational data model for answering imprecise query against precise data. We define the fuzzy relation which is an extension of the standard relation. Two formal languages, a fuzzy tuple relational calculus and a fuzzy relational algebra, are provided for specifying the imprecise queries. Examples are given to show the wide range of imprecise queries expressible by the given languages. The equivalence of the two formal languages has been proved. We also discuss the techniques that can be used to implement a fuzzy database system based on the proposed fuzzy relational model.

We are currently developing a fuzzy relational database system based on the ideas presented in this paper. An interesting issue for future study is the query optimization that involves many fuzzy relations, both infinite and finite. Another research issue is to build tools to help users to define fuzzy relations that best suits their needs. We are also interested in further extending our model to include various fuzzy data.

# REFERENCES

[1] P. Bosc, M. Galibourg and G. Hamon, "Fuzzy quering with SQL: Extensions and implementation aspects", *Fuzzy Sets and Systems*, Vol. 28, 1988, pp. 333-349.

[2] B. P. Buckles and F. E. Petry, "A fuzzy representation of data for relational databases", *Fuzzy Set and Systems*, Vol. 7, No. 3, 1982, pp. 213-226.

[3] B. P. Buckles and F. E. Petry, "Fuzzy databases and their applications", in *Fuzzy Information and Decision Process*, M. M. Gipta and E. Sanchez, Eds., North Holland, 1982, pp. 361–371.

[4] B. P. Buckles and F. E. Petry, "Information-theoretic characterization of fuzzy relational databases", *IEEE Trans. System Man Cybernet*, vol. SMC-13, No. 1, 1983, pp. 74–77.

[5] B. P. Buckles and F. E. Petry, "Extending the fuzzy database with fuzzy numbers", *Information Sciences*, Vol. 34, No. 2, 1984, pp. 145–155.

[6] S. K. Chang and J. S. Ke, "Translation of fuzzy queries for relational database systems", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 1, 1979, pp. 281–294.

[7] C. J. Date, *An Introduction to Database Systems*, two volumes, Addison-Wesley, Readings, MA, 1986.

[8] J. Kacprzyk, S. Zadrozny, and A. Ziolkowski, "FQUERY III+, a "human-consistent" database querying system based on fuzzy logic with linguistic quantifiers", *Information Systems*, Vol. 14, No. 6, 1989, pp. 443–453.

[9] J. Kacprzyk and A. Ziolkowski, "Data base queries with fuzzy linguistic quantifiers, *IEEE Trans. Systems, Man and Cybernetics*, Vol. 16, No. 3, 1986, pp. 474–478.

[10] J. Kacprzyk and A. Ziolkowski, "Queries with fuzzy linguistic quantifiers using an alternative calculus of linguistically quntified propositions", *Proc. Second IFSA Congress*, Tokyo, Japan, 1987, pp.600–603.

[11] H. F. Korth and A. Silberschatz, *Database System Concepts*, Second Edition, McGraw-Hill, New York, NY, 1991.

[12] D. Li and D. Liu, *A Fuzzy Prolog Database System*, Research Studies Press, Taunton, England, 1990.

[13] *Fuzzy LUNA — Fuzzy Database System Library User's Manual*, and *Fuzzy LUNA — Fuzzy Database System Library Reference Manual*, OMRON Corparation, 1992.

[14] S. Shenoi and A. Melton, "An Extended Version of the Fuzzy Relational Database Model", *Information Sciences*, Vol. 52, 1990, pp. 35–52.

[15] V. Tarani, "A conceptual framework for fuzzy query processing — a step towards very intelligent database systems", *Information Processing and Management*, Vol. 13, 1977, pp. 289–303.

[16] J. D. Ullman, *Principles of Database Systems*, Computer Science Press, Rockville, MD, 1983.

[17] J. D. Ullman, *Principles of Database and Knowledge-Base Systems*, Vol. I, Computer Science Press, Rockville, MD, 1988.

[18] L. A. Zadeh, "Fuzzy set", *Information and Control*, Vol. 8, 1965, pp.338–353.

[19] L. A. Zadeh, "Fuzzy Logic", *IEEE Computer*, April, 1988, pp. 83–93.

[20] M. Zemankova and A. Kandel, "Implementing imprecision in information systems", *Information Sciences*, Vol. 37, 1985, pp. 107–141.

[21] W. Zhang, C. Yu, G. Wang, T. Pham, and H. Nakajima, "A Relational Model for Imprecise Queries", Technical Report, Dept. of EECS, Univ. of Illinois at Chicago, Chicago, IL., 1992.

## INTERNAL DISTRIBUTION

1. B. R. Appleton
2. J. E. Baker
3. A. L. Bangs
4. M. Beckerman
5. R. J. Carter
6. O. H. Doerum
7. J. R. Einstein
8. C. W. Glover
9-13. K. S. Harber
14. J. P. Jones
15. H. E. Knee
16. R. C. Mann
17. E. M. Oblow
18. F. G. Pin
19. S. A. Raby

20. K. Rahmani
21. D. B. Reister
22. J. C. Schryver
23. P. F. Spelt
24. E. C. Uberbacher
25. M. A. Unseren
26. R. C. Ward
27-28. Laboratory Records
       Department
29. Laboratory Records,
    ORNL-RC
30. Document Reference
    Section
31. Central Research Library
32. ORNL Patent Section

## EXTERNAL DISTRIBUTION

33. Dr. Peter Allen, Department of Computer Science, 450 Computer Science, Columbia University, New York, NY 10027

34. Dr. Wayne Book, Department of Mechanical Engineering, J. S. Coon Building, Room 306, Georgia Institute of Technology, Atlanta, GA 30332

35. Professor Roger W. Brockett, Wang Professor of Electrical Engineering and Computer Science, Division of Applied Sciences, Harvard University, Cambridge, MA 02138

36. Dr. Steven Dubowsky, Massachusetts Institute of Technology, Building 3, Room 469A, 77 Massachusetts Ave., Cambridge, MA 02139

37. Professor Donald J. Dudziak, Department of Nuclear Engineering, 110B Burlington Engineering Labs, North Carolina State University, Raleigh, NC 27695-7909

38. Dr. Avi Kak, Department of Electrical Engineering, Purdue University, Northwestern Ave., Engineering Mall, Lafayette, IN 47907

39-188. Professor Dr. Jan Komorowski, Knowledge Systems Group, Faculty of Computer Science and Electrical Engineering, The Norwegian Institute of Technology, The University of Trondheim, O.S. Bragstads plass 2E, N-7034 Trondheim, NORWAY

189. Dr. James E. Leiss, Rt. 2, Box 142C, Broadway, VA 22815-9303

190. Dr. Oscar P. Manley, Division of Engineering, Mathematical, and Geosciences, Office of Basic Energy Sciences, ER-15, U.S. Department of Energy - Germantown, Washington, DC 20545

191. Professor Neville Moray, Department of Mechanical and Industrial Engineering, University of Illinois, 1206 West Green St., Urbana, IL 61801

192. Dr. Zbigniew Ras, University of North Carolina at Charlotte, Department of Computer Science, Kennedy Bldg., Charlotte, NC 28223

193. Dr. Wes Snyder, Department of Radiology, Bowman Gray School of Medicine, 300 S. Hawthorne Dr., Winston-Salem, NC 27103

194. Professor Mary F. Wheeler, Department of Mathematical Sciences, Rice University, P.O. Box 1892, Houston, TX 77251

167

195. Office of Assistant Manager for Energy Research and Development, U.S. Department of Energy, Oak Ridge Operations Office, P.O. Box 2001, Oak Ridge, TN 37831-8600

196–197. Office of Scientific Technical Information, P.O. Box 62, Oak Ridge, TN 37831